THE MICROSOFT JOURNAL FOR DEVELOPERS

APRIL 2010 VOL 25 NO 4

msdn magazine

VISUAL STUDIO 2010

COLUMNS

This month at msdn.microsoft.com/magazine:

USABILITY IN PRACTICE
The Decade of Convergence
Charles B. Kreitzberg

Microsoft

msdn® magazine

APRIL 2010 VOLUME 25 NUMBER 4

MSDN Magazine (ISSN 1528-4859) is published monthly by 1105 Media, Inc., 9201 Oakdale Avenue, Ste. 101, Chatsworth, CA 91311. Periodicals postage paid at Chatsworth, CA 91311-9998, and at additional mailing offices. Annual subscription rates payable in U.S. funds: U.S. $35; Canada $45; International $60. Single copies/back issues: U.S. $10, all others $12. Send orders with payment to: MSDN Magazine, PO. Box 3167, Carol Stream, IL 60132, e-mail MSDNmag@1105service.com or call 847-763-9560. **POSTMASTER:** Send address changes to *MSDN Magazine*, PO. Box 2166, Skokie, IL 60076. Canada Publications Mail Agreement No: 40612608. Return Undeliverable Canadian Addresses to Circulation Dept. or IMS/NJ. Attn: Returns, 310 Paterson Plank Road, Carlstadt, NJ 07072.

Your best source for software development tools!

programmer's paradise

LEADTOOLS Recognition SDK v16.5

by LEAD Technologies

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

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

Paradise # L05 26301A01

$3,214.99

programmers.com/LEAD

Multi-Edit 2008

by Multi Edit Software

Multi-Edit 2008 delivers, a powerful IDE, with its speed, depth, and support for over 50 languages. Enhanced search functions include Perl 5 Regular Expressions and definable filters. Supports large DOS/Windows, UNIX, binary and Mac files. File Sync Integration for: Delphi 6, 7, 2005, C++ Builder 6, BDS 2006 and Rad Studio 2007, VB 6, VC 6, VS 2003, 2005 and 2008. Includes file compare, code beautifying, command maps, and much more.

1-49 Users
Paradise # A30 01201A01

$179.99

programmers.com/multiedit

VMware vSphere

Put time back into your day.

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

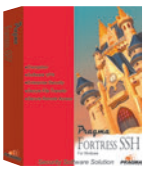
- Increase the productivity of your existing staff three times over
- Control downtime—whether planned or not
- Save more than 50% on the cost of managing, powering and cooling servers

Make your time (and money) count for more with virtualization from VMware.

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

$9,234.99

programmers.com/vSphere



Certified for Windows 7/2008R2

Pragma Fortress SSH—SSH Server & Client for Windows

by Pragma Systems

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

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

Paradise # P35 04201A01

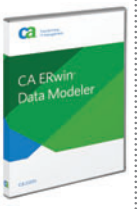
$550.99

programmers.com/pragma

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

by CA

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



Paradise # P26 04201E01

$3,951.99

programmers.com/ca

BUILD ON VMWARE ESXi AND VSPHERE

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

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

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

NEW VERSION 6!

ActiveReports 6

by GrapeCity

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

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

Professional Ed.
Paradise # D03 04301A01

$1,310.99

programmers.com/grapecity

Orion Network Performance Monitor

by Solarwinds

Orion Network Performance Monitor is a comprehensive fault and network performance management platform that scales with the rapid growth of your network and expands with your network management needs. It offers out-of-the-box network-centric views that are designed to deliver the critical information network engineers need. Orion NPM is the easiest product of its kind to use and maintain, meaning you will spend more time actually managing networks, not supporting Orion NPM.

Paradise # S4A 08201E02

$4,606.99

programmers.com/solarwinds

TX Text Control 15.1

Word Processing Components

NEW RELEASE!

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

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

Professional Edition
Paradise # T79 02101A02

$848.99

Download a demo today.

programmers.com/theimagingsource

AdminStudio®

AdminStudio & Application Virtualization Pack

by Flexera Software

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

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

$4,228.99

programmers.com/flexera

DON'T BE LEFT BEHIND! STAY ON THE CUTTING EDGE OF TECHNOLOGY: NEW! MICROSOFT® VISUAL STUDIO® 2010 MAKES IT EASY!

Microsoft® Visual Studio® 2010

- **Set your ideas free**—Create what you can imagine, build on the strengths of your team, and open up new possibilities!
- **Simplicity through integration**—A single integrated development environment that takes your skills further and adjusts to the way you work.
- **Quality tools help ensure quality results**—Powerful testing tools with proactive project management features help you build the right app the right way.

DISCOVER MICROSOFT VISUAL STUDIO 2010...
TAKE YOUR DEVELOPMENT TEAM TO THE NEXT LEVEL!

Call your Programmer's Paradise Representative Today!

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

Prices subject to change. Not responsible for typographical errors.

EDITOR'S NOTE

KEITH WARD

Scott Guthrie on Visual Studio 2010

These are whirlwind days for Microsoft's Scott Guthrie. A corporate vice president for the .NET developer platform, he oversees development of Visual Studio, the Microsoft .NET Framework, ASP.NET, Silverlight, CLR and more. Many of those products are in the "milestone" phase, with major new versions being released, or about to be released. That includes, of course, Visual Studio 2010 and the .NET Framework 4, which take flight in April. Silverlight 4 is on the way, too.

And you thought your job was busy.

Because this issue covers the launch of Visual Studio 2010, it made sense to talk to the man most responsible for getting it out the door. And Guthrie had much to say.

Visual Studio 2010, he says, "is a pretty big release. From a feature set and capability set, it's pretty ambitious." The goal from the beginning, Guthrie explains, was not just to create new features, but to answer the question: "How do we make existing developers' lives easier?" To that end, he says, Visual Studio 2010 includes "so many new features that don't require you to learn a whole bunch of new things."

Using multiple monitors is one example. It's something that isn't a knock-'em-dead upgrade, but can definitely enhance productivity. Guthrie explains why it excites him: "Most developers have a multimonitor setup at work. Visual Studio in the past didn't let you leverage those monitors. In Visual Studio 2010, they can tear off any code-editing window and drop it into the second monitor. It makes navigating through large products a lot easier."

There's even more stuffed into Visual Studio 2010. Silverlight, for instance, is fully supported for the first time—and Guthrie says Silverlight 4 will be as well, as soon as it ships. Visual Studio 2010 also marks the debut of F#, a new .NET Framework-based language for large-scale parallel programming, such as that done in scientific and financial settings.

It's those kinds of enhancements that, when taken as a whole, will make developers more productive. Guthrie puts it this way: "[There are] a lot of things that people go 'Finally! Yes, great, I've been asking for that!'"

Guthrie also notes that making sure Visual Studio 2010 was stable and fast out the door led to the month delay in shipping; as you may remember, it was originally scheduled to ship in March. User feedback on the beta resulted in the slippage, he says: "We got feedback that performance and stability weren't where they should be. We moved the dates to make sure we had confidence that we were building the right product."

Guthrie believes Microsoft succeeded. Visual Studio 2010, he says, "makes developers much more productive and writing code a lot more fun."

Magazine Revisions

It's not only Visual Studio 2010 and .NET that are getting upgraded this month. *MSDN Magazine* has experienced a number of changes. To begin with, we've incorporated the new Microsoft MSDN logo as the nameplate on the cover. The cover had not been updated in some time, and it's always good to look at things in a new light. Plus, the "network wave" is way cool, I think. The more "artsy" readers may also notice that our color palette has been modified to align with the logo colors. It makes the magazine feel more cohesive.

A more subtle change on the cover is that we've increased the font size of the text. MSDN sports a text-heavy cover, and making it a little easier to read all that text is what we're going for here.

Inside, the most immediate change you'll notice is that we've added illustrations for our regular columnists like Charles Petzold, Dino Esposito, David Platt and others. These folks are our core contributors, and I hope that by seeing their images, you might feel a little more connected to them. After all, coders are people too!

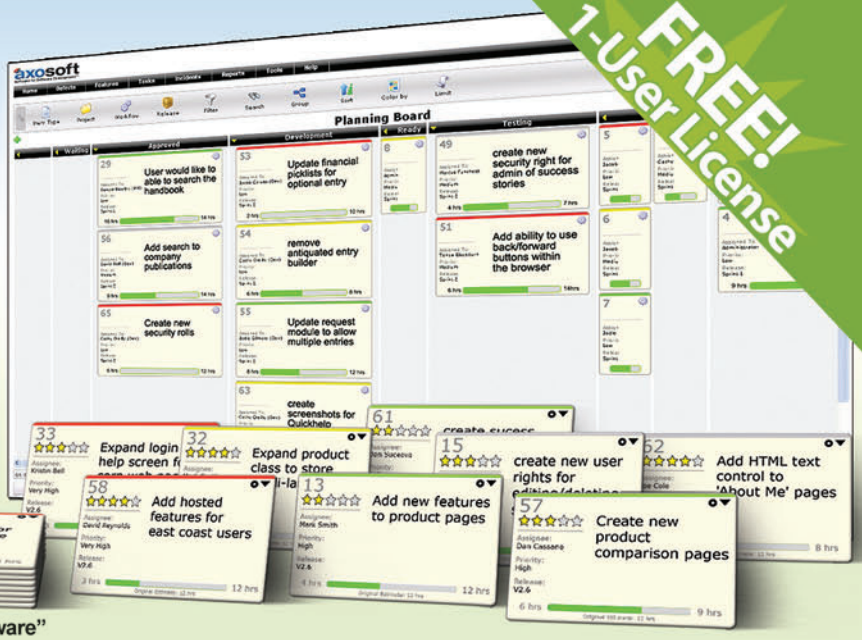
If you're using Visual Studio 2010, we'd love to hear from you on what you like—and don't like—about it. We also want your feedback on the changes in the magazine—do they work for you? Should we have left well-enough alone? What other changes would you like to see? Send all comments to mmeditor@microsoft.com.

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

Introducing OnTime 2010

NEW! Scrum Planning Board that Changes Everything!

FREE! 1-User License



Reader's Choice 2006 aspnet Winner · Reader's Choice 2007 aspnet Winner · Reader's Choice 2008 aspnet Winner · Reader's Choice 2009 aspnet PRO Winner

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

Project Management • Bug Tracking • Scrum

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

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

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

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

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

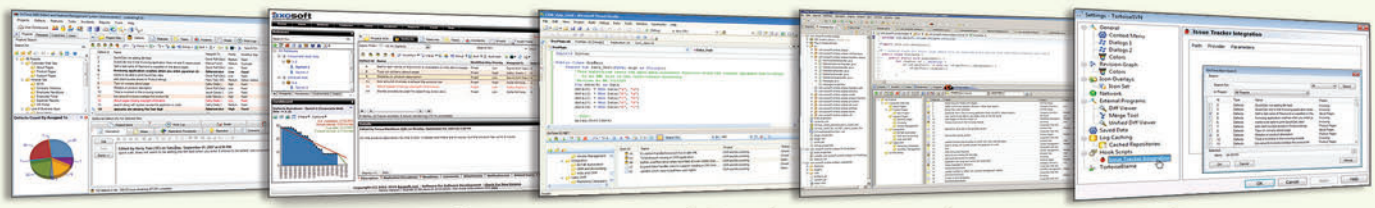
Workflow
Define a hyper-fast, lightweight, agile workflow or a robust process-enforcement system.

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

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

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

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



Windows *Web* *Visual Studio* *Eclipse* *Sub-Version*

axosoft

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

800.653.0024
www.axosoft.com

DINO ESPOSITO

Revisiting Asynchronous ASP.NET Pages

ASP.NET has always supported synchronous and asynchronous HTTP handlers. Now, ASP.NET 2.0 has new features to make it easier and quicker for developers to create asynchronous pages. Especially for server-based applications, asynchronous operations are fundamental for enabling scalability. If you find you need to scale up your existing Web application, the first aspect to consider is how much asynchrony you can add to the pages.

In this regard, ASP.NET behaves like any other server application that performs some background work on behalf of multiple clients. Each incoming request is assigned to an ASP.NET-owned thread that is picked up from the ASP.NET thread pool. The thread remains blocked until the operation has terminated and some response has been generated for the client. How long should the thread wait? The ASP.NET runtime environment can be configured to define a custom timeout (90 seconds is the default), but it's more important to prevent the thread from being blocked.

When you deal with potentially lengthy operations, the timeout at most only ensures that after a given number of seconds, the thread will be freed and returned to the pool. Instead, what you want is to keep the thread from being blocked for a long time. Ideally, you want the thread to begin a request and then yield it to some other non-ASP.NET thread. The same thread, or another one from the ASP.NET pool, will be picked up again upon completion of the operation to send the response to the client. This paradigm is known as asynchronous ASP.NET pages.

> You should distinguish between pages that are asynchronous with respect to the user and pages that are asynchronous with respect to the ASP.NET runtime.

When it comes to asynchronous operations, you should distinguish between pages that are asynchronous with respect to the user and pages that are asynchronous with respect to the ASP.NET runtime. For pages asynchronous with respect to the user, the only viable approach is an AJAX operation. However, using AJAX to perform a potentially slow operation diminishes the impact on the end user but doesn't bring any relief to the ASP.NET runtime.

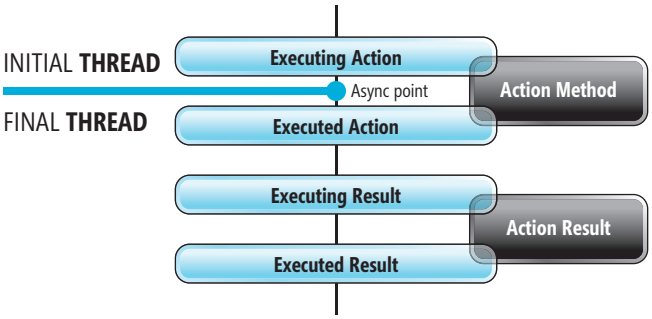


Figure 1 **Mechanics of an Async Action Method in ASP.NET MVC**

Async Pages and the ASP.NET Runtime

The longer the thread hangs onto the request, the longer one thread is subtracted from the ASP.NET pool to serve new incoming requests. When no threads are available to serve new requests, the requests are queued. This may lead to delays and degradation of overall performance.

In ASP.NET, HTTP handlers are synchronous by default. Asynchronous HTTP handlers must be explicitly architected and implemented by applying slightly different interfaces. A synchronous handler differs from an asynchronous handler in one key aspect: instead of the synchronous ProcessRequest method, an asynchronous handler uses the methods listed below, which are part of the IHttpAsyncHandler interface:

```
IAsyncResult BeginProcessRequest(
     HttpContext context,
     AsyncCallback cb,
     object extraData);

void EndProcessRequest(
     IAsyncResult result);
```

BeginProcessRequest contains the operation to be executed to service the request. This code should be designed to start the operation on a secondary thread and return immediately. EndProcessRequest contains the code to complete the request that was previously started.

This column discusses a prerelease version of ASP.NET MVC 2. All information is subject to change.

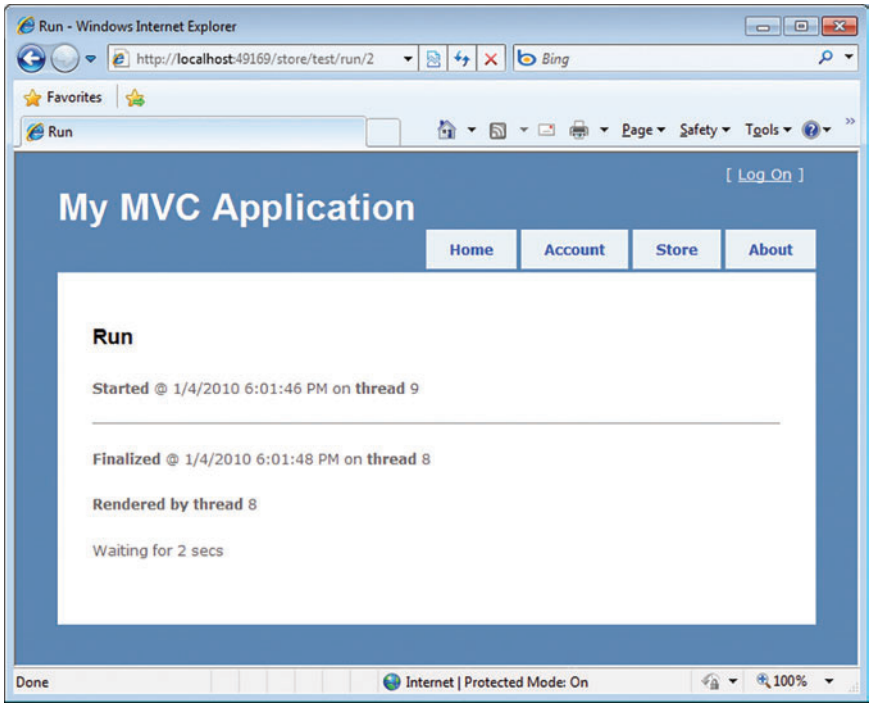


Figure 2 **Thread Switching for an Async Action Method Call**

As you can see, an asynchronous HTTP request is split into two parts—before and after the "async point"—the point in the request lifecycle where the thread that owns the request changes. When the async point is reached, the original ASP.NET thread yields control to another thread. This potentially lengthy operation takes place in between the two parts of the ASP.NET request. Each part of the async request runs independent of the other, with no affinity as far as the thread is concerned. In other words, there's no guarantee that the same thread will take care of the two parts of the request. The net effect is that no threads are blocked for the duration of the operation.

> Using AJAX to perform a potentially slow operation diminishes the impact on the end user but doesn't bring any relief to the ASP.NET runtime.

At this point, the obvious question is: which thread really takes care of the "lengthy" operation? ASP.NET uses I/O completion ports internally to track the termination of a request. When the async point is reached, ASP.NET binds the pending request to an I/O completion port and registers a callback to get a notification when the request has terminated. The OS will use one of its own dedicated threads to monitor the termination of the operation, thus freeing the ASP.NET thread from having to wait in full idle. When the operation terminates, the OS places a message in the completion queue, which triggers the ASP.NET callback that will then pick up one of its own threads to resume the request. As mentioned, I/O completion ports are a feature of the OS.

The Real Nature of Async Pages

In ASP.NET, async pages are commonly associated with the idea of improving the performance of a given page in charge of performing a potentially lengthy operation. However, a few further points should be noted. From the user's perspective, synchronous and asynchronous requests look nearly the same. If the requested operation is expected to take, say, 30 seconds to complete, the user will wait at least 30 seconds to get the new page back. This happens regardless of the synchronous or asynchronous implementation of the page. Also, don't be too surprised if an async page ends up taking a bit more time to complete a single request. So what's the benefit of asynchronous pages?

Scalability is not quite the same as performance. Or, at least, scalability is about performance but on a different level—the whole application instead of a single request. The benefit that async pages bring to the table is much less work for the threads in the ASP.NET pool. This doesn't make long requests run faster, but it helps the system serve non-lengthy requests as usual—that is, with no special delays resulting from ongoing slow requests.

Async requests take advantage of asynchronous HTTP handlers, which have always been a feature of the ASP.NET platform. However, both ASP.NET Web Forms and ASP.NET MVC provide their own facilities to make it simpler for developers to implement async actions. In the remainder of the article, I'll discuss asynchronous operations in ASP.NET MVC 2.

Asynchronous Controller Actions

In ASP.NET MVC 1.0, any controller action can only run synchronously. However, a new AsyncController class was added to the MVC Futures library. After an experimental period, the async API for controllers was officially added to the ASP.NET MVC framework, and it's fully available and documented as of version 2 of the ASP.NET MVC framework. (The syntax and features discussed in this article refer to ASP.NET MVC 2 RC.) If you play a bit with the AsyncController class in the MVC Futures library, you will notice some changes, and the API is simpler and cleaner.

The purpose of the AsyncController is to ensure that any exposed action methods execute asynchronously without changing the overall approach to programming that characterizes the ASP.NET MVC framework. The diagram in **Figure 1** shows the sequence of steps behind the processing of an async action.

The async point is placed in between the executing and executed events. When the action invoker notifies that it's about to execute

the action, the thread engaged is still the original ASP.NET thread that picked up the request from the Web server queue. At this point, the action is executed. At the end, when the action invoker is ready to notify the action-executed event, possibly another ASP.NET thread is taking care of the request. **Figure 2** illustrates this scenario.

Before I discuss the details of how you create and debug asynchronous methods, another fundamental point of asynchronous ASP.NET operations should be made clear: not all actions are good candidates for becoming async operations.

> The benefit that async pages bring to the table is much less work for the threads in the ASP.NET pool.

The Real Target of Async Operations

Only I/O-bound operations are good candidates for becoming async action methods on an asynchronous controller class. An I/O-bound operation is an operation that doesn't depend on the local CPU for completion. When an I/O-bound operation is active, the CPU just waits for data to be processed (that is, downloaded) from external storage (a database or a remote service). I/O-bound operations are in contrast to CPU-bound operations, where the completion of a task depends on the activity of the CPU.

A typical example of an I/O-bound operation is the invocation of a remote service. In this case, the action methods fire the request and then just wait for any response to be downloaded. The real work is being done remotely by another machine and another CPU. Thus, the ASP.NET thread is stuck waiting and being idle. Releasing that idle thread from the duty of waiting to serve other incoming requests is the performance gain you can achieve with async implementation of actions or pages.

It turns out that not all lengthy operations will give you a concrete benefit if implemented asynchronously. A lengthy in-memory calculation doesn't significantly benefit from asynchronous implementation. It could even run slightly slower, because the same CPU is serving both the ASP.NET request and the calculation. In addition, you may still need an ASP.NET thread to physically take care of the calculation. There's little benefit, if any, in using the async implementation for CPU-bound operations. On the other hand, if remote resources are involved, even multiple resources, using async methods can really boost the performance of the application, if not the performance of the individual request.

I'll return to this point shortly with an example. For now, let's focus on the syntax required to define and execute async actions in ASP.NET MVC.

Recognizing Async Routes

In which way is an async route different from a synchronous route? In MVC Futures, you were asked to use different methods to register

synchronous and asynchronous routes. Here's the old way to register an async route:

```
routes.MapAsyncRoute(
    "Default",
    "{controller}/{action}/{id}",
    new { controller = "Home", action = "Index", id = "" }
);
```

You had to use the MapAsyncRoute extension method instead of the standard MapRoute you would have used for classic synchronous methods. In ASP.NET MVC 2 RC, however, this distinction has been removed. Now you have just one way to register your routes—the MapRoute method—regardless of how the action will then be executed.

The URL of the request is therefore processed as usual and the name of the controller class to use is figured out. It is required, in fact, that an async method is defined on a controller class that derives from the new AsyncController class, illustrated here:

```
public class TestController : AsyncController
{
  ...
}
```

If the controller class inherits from AsyncController, the convention for mapping action names to methods is a bit different. An AsyncController class can serve both synchronous and asynchronous requests. As a result, the convention used can recognize both a method Run and a method RunAsync, as shown here:

```
public class TestController : AsyncController
{
  public ActionResult Run(int id)
  {
    ...
  }
  public void RunAsync(int id)
  {
    ...
  }
}
```

If you do this, however, an exception will be thrown (see **Figure 3**).

An async action is identified by name, and the expected pattern is xxxAsync, where xxx indicates the default name of the action to execute. Clearly, if another method named xxx exists, and it is not disambiguated using attributes, then an exception is thrown as in **Figure 3**.

The word Async is considered a suffix. The URL to invoke the RunAsync method will contain only the prefix Run. For example, the following URL will invoke the method RunAsync, passing a value of 5 as a route parameter:

```
http://myserver/demo/run/5
```

ASP.NET Web Forms and ASP.NET MVC offer higher-level tools to code async operations, each within their own application model.

Whether this will be resolved as a synchronous or asynchronous action depends on the methods you have in the AsyncController class. The xxxAsync method, however, identifies only the trigger of the operation. The finalizer of the request is another method in the controller class named xxxCompleted:

```
public ActionResult RunCompleted(DataContainer data)
{
    ...
}
```

Note the different signature of the two methods defining the async action. The trigger is expected to be a void method. If you define it to return any value, the return value simply will be ignored. The input parameters of the xxxAsync method will be subject to model binding as usual. The finalizer method returns an ActionResult object as usual and it receives a custom object that contains the data it's expected to process and pass on to the view object. A special protocol is necessary for matching the values calculated by the trigger to the parameters declared by the finalizer.

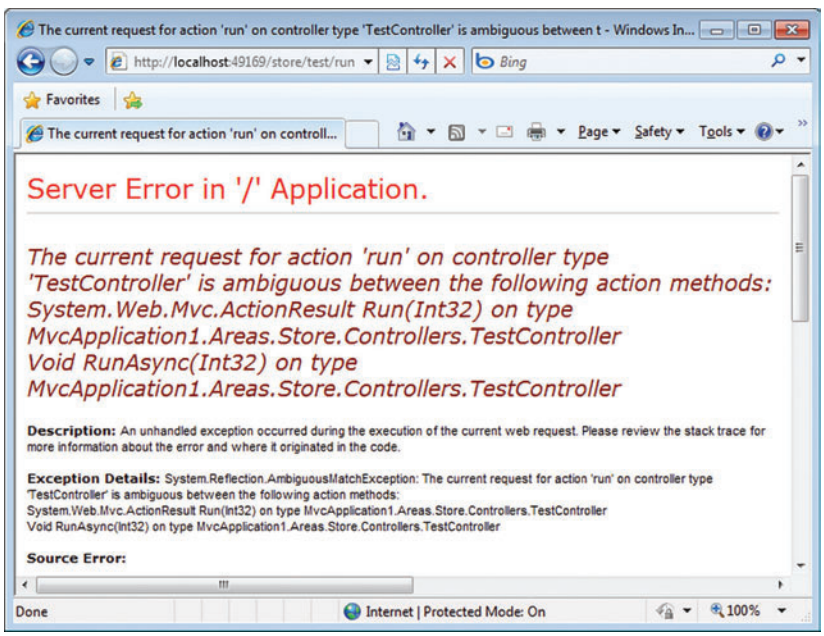


Figure 3 **Ambiguous References in the Name of the Action**

The AsyncController Class

The AsyncController controller class inherits from Controller and implements a bunch of new interfaces as shown here:

```
public abstract class AsyncController : Controller,
                IAsyncManagerContainer,
IAsyncController, IController
```

The most distinctive aspect of an async controller is the special action invoker object that is employed under the hood to perform operations. The invoker needs a counter to track the number of individual operations that compose the action and that must be synchronized before the overall action can be declared terminated. **Figure 4** provides a sample implementation for an async action.

The OutstandingOperations member on the AsyncManager class provides a container that maintains a count of pending asynchronous

Figure 4 **A Simple Asynchronous Action Method**

```
public void RunAsync(int id)
{
    AsyncManager.OutstandingOperations.Increment();

    var d = new DataContainer();
     ...

    // Do some remote work (i.e., invoking a service)
     ...

    // Terminate operations
    AsyncManager.Parameters["data"] = d;
    AsyncManager.OutstandingOperations.Decrement();
}
public ActionResult RunCompleted(DataContainer data)
{
   ...
}
```

operations. It's an instance of the OperationCounter helper class and supplies an ad hoc API to increment and decrement. The Increment method is not limited to unary increments, as shown here:

```
AsyncManager.OutstandingOperations.Increment(2);
service1.GetData(...);
AsyncManager.OutstandingOperations.Decrement();
service2.GetData(...);
AsyncManager.OutstandingOperations.Decrement();
```

The AsyncManager Parameters dictionary is used to group values to be passed as arguments to the finalizer method of the async call. The Parameters dictionary is expected to contain an entry for each parameter to be passed to the finalizer—the xxxCompleted method in the earlier example. If a match can't be found between entries in the dictionary and parameter names, a default value is assumed for the parameter—null for reference types. No exception is raised unless an attempt is made to access a null object. The xxxCompleted method receives parameters of any supported type and uses them to fill up the ViewData collection or any strong-typed object recognized by the view. The xxxCompleted method is responsible for returning an ActionResult object.

A Good Fit or Not?

Wrapping up, synchronous requests are a necessary feature in ASP.NET and, in fact, asynchronous HTTP handlers have been supported since ASP.NET 1.0.

ASP.NET Web Forms and ASP.NET MVC offer higher-level tools to code async operations, each within their own application model—in ASP.NET MVC, you have async controllers, and in Web Forms you rely on async pages.

The key aspect of async actions, though, is deciding whether a given task is a good fit for an asynchronous implementation. Async methods should only be built around I/O-bound operations. And, finally, bear in mind that async methods won't run faster themselves, but will allow other requests to run faster. ■

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

Thanks to the following technical expert for reviewing this article: *Stefan Schackow*

Better Coding with Visual Studio 2010

Doug Turnure

It's been 13 years since Microsoft first rolled out Visual Studio, its long-running flagship integrated development environment. The 1997 inaugural release began the alignment of versions 5 of Visual Basic and Visual C++ into a common IDE. It included an Enterprise Edition with a once-famous cast of technologies including Visual InterDev, Microsoft Transaction Server and Visual SourceSafe 5.0. The core purpose of that release was to help developers build distributed applications with components, both as client/server and as Web applications.

We've come a long way as developers since then. While modified forms of client/server and Web architectures still dominate the coding scene, the scope and demands of these applications have exploded out of the realm of homogenous systems and strongly typed object-to-object calls.

Representational State Transfer (REST) and related technologies are now becoming mainstream foundational communication mechanisms. Applications are beginning to find a home in cloud-based architectures, putting scalability, reliability and data security in the hands of third parties. And Web application patterns are aligning to commoditized styles and standards. Even hardware is changing, as processor speeds are nearing theoretical peaks with current chip technology, and multicore systems are providing the new way to squeeze ever more performance out of a single computer.

It is into this world, and its unprecedented demands of software and software developers, that Visual Studio 2010 arrives. And, loaded with new capabilities, it stands poised to deliver on the growing requirements of software in a world that now stretches across platforms, cores, styles and standards. This article will cover some of the prominent ways Visual Studio 2010 addresses today's challenges, to help developers build solutions that thrive in the complex industry where they now reside. Not every new capability will be covered here; rather, the goal of this article is to highlight some of the specific features that developers will be able to put to use immediately in their daily work. Full feature lists are readily available in the product documentation.

This article discusses a prerelease version of Visual Studio 2010 and the Microsoft .NET Framework 4. All information is subject to change.

This article discusses:

- Visual Studio IDE improvements
- Parallel programming
- Web development
- MVC integration
- SharePoint integration
- Windows 7 development
- Windows Azure integration
- Multi-targeting

Technologies discussed:

Visual Studio 2010, Microsoft .NET Framework 4, Multicore Architecture, Parallel Programming, ASP.NET AJAX, Model-View-Controller Architecture, SharePoint, Windows 7, Windows Azure Platform

Visual Studio IDE Improvements

As software development continues to address more and more complex scenarios, developers often feel pressure to be more productive. Visual Studio 2010 adds several new visual features to the editor itself that help with productivity. I'll discuss three of them in this section.

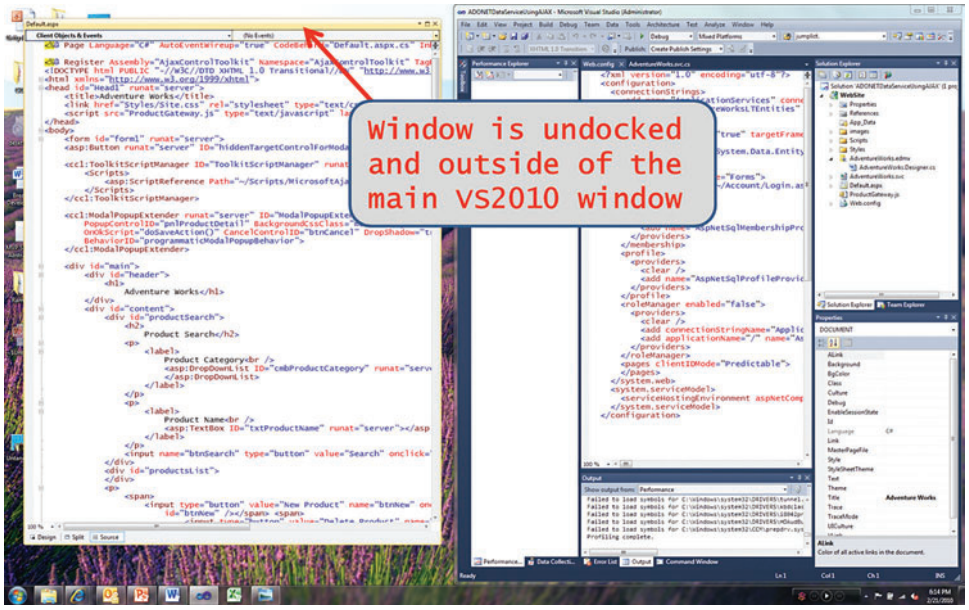


Figure 1 **Untethering a Window from the IDE**

One of the simpler improvements to try out in the Visual Studio 2010 IDE is the ability to drag a child window outside of the IDE parent window. Monitors have become affordable enough to allow developers to use two (or more) in their work, and this new feature lets you spread out your coding and design windows across multiple screens.

How It Works *Simply click on the tab for the window and drag it outside the Visual Studio IDE window. The window will re-dock just as easily by clicking the title bar and dragging it back into the IDE area, aligning the title bar to the tabs of the other windows. When re-docking, you'll receive a visual cue when the window is properly aligned: the window will become shaded to indicate it's ready to re-dock when you release it.* **Figure 1** *shows a code window that has been pulled outside of the IDE's containing window.*

Another nice new feature in the Visual Studio 2010 editor is the ability to box select and edit a vertical block of text. Sometimes you need to apply edits to an extended list of items, and you find yourself doing the familiar rapid-repeat keyboard sequence (for example, "n+down arrow+back arrow" over and over). While this may not be the most frequent problem you face, it does happen occasionally, and many developers find themselves wishing they could select a whole vertical block and apply a common change to all lines at once. Visual Studio 2010 introduces this block-edit capability.

How It Works *Press and hold down the Alt key and make the vertical selection you want to edit. The highlighted area will function just like a single-line edit, but the edits will be duplicated simultaneously on all selected lines, as shown in* **Figure 2**.

There is also a helpful highlighting feature for references in your code. In the editor, when you click on a variable, object, method, property, type or other symbol, all references to that symbol in your code will become highlighted to help you quickly discover where the item is being used.

How It Works *Pick a type/variable/method/whatever and simply click on it, and the other instances will become highlighted.* **Figure 3** *shows this for a variable named jumpList; note the gray shading of all uses in code after selecting it with a mouse click.*

```
int iBackColor;
int iForeColor;
int iStyle;
int iSize;
int iShape;
```

Figure 2 **Box Selecting**

Support for Parallel Programming

Moving a bit deeper into the new IDE capabilities, the next significant new feature I'll cover is support for parallel programming. Visual Studio 2010 ships with diagnostic tools to help debug and analyze parallel applications. Before jumping into the tooling, however, I'll briefly discuss what parallel programming is, and then we can see how the runtime and libraries implement these new capabilities.

Many of today's standard developer machines are multicore, meaning they have two or more individual processors, and the current trend is toward machines with many more cores. Aside from pure innovation, factors such as power conservation and realistic limitations of clock-speed, power consumption and heat are influencing the trend toward multicore systems. Some industry leaders predict that mainstream machines will have 50 or more cores in the next few years.

One of the simpler improvements to try out in the Visual Studio 2010 IDE is the ability to drag a child window outside of the IDE parent window.

This presents a unique problem. Prior to the multicore revolution, faster machines meant faster-running applications. However, software that is tied to a single core (as is most current software) can't take advantage of this emerging architecture. It is therefore critical to target multicore architectures going forward. Before Visual Studio 2010 and the .NET Framework 4, writing code that could make use of more than one core was difficult. To meet this challenge, this release includes some updates to the runtime—and some new types, libraries and tooling—to help developers take advantage of multicore systems.

The .NET Framework 4 includes Parallel Extensions, which has three components: a new

```
TaskbarDemoMainForm.cs
TaskbarDemo.TaskbarDemoMainForm | buttonUserTasksAddTasks_Click(object sender, EventArgs e)

        private void buttonUserTasksAddTasks_Click(object sender, EventArgs e)
        {
            // Path to Windows system folder
            string systemFolder = Environment.GetFolderPath(Environment.SpecialFolder.System);

            // Add our user tasks
            jumpList.AddUserTasks(new JumpListLink(Path.Combine(systemFolder, "notepad.exe"), "Open Notepad")
            {
                IconReference = new IconReference(Path.Combine(systemFolder, "notepad.exe"), 0)
            });

            jumpList.AddUserTasks(new JumpListLink(Path.Combine(systemFolder, "mspaint.exe"), "Open Paint")
            {
                IconReference = new IconReference(Path.Combine(systemFolder, "mspaint.exe"), 0)
            });

            jumpList.AddUserTasks(new JumpListSeparator());

            jumpList.AddUserTasks(new JumpListLink(Path.Combine(systemFolder, "calc.exe"), "Open Calculator")
            {
                IconReference = new IconReference(Path.Combine(systemFolder, "calc.exe"), 0)
            });

            // Update status
            UpdateStatusBar("Three user tasks added to jump list");
        }
100 %
```

Figure 3 **Highlighted Symbols**

Task Parallel Library (TPL), a new PLINQ Execution Engine and a handful of new Coordination Data Structures (CDS). The TPL contains two primary classes. One of these, System.Threading.Tasks.Parallel, includes parallel constructs, such as parallel versions of For and ForEach methods. As the name implies, the PLINQ Execution Engine is a parallelized version of LINQ to Objects, providing a ParallelEnumerable class in place of LINQ's Enumerable class. Using PLINQ is similar, but not identical, to using LINQ to Objects. Finally, CDS includes a group of thread-safe collections and synchronization primitives to simplify parallel scenarios. CDS includes the familiar players you'd expect in thread-safe collections (ConcurrentDictionary, ConcurrentQueue and ConcurrentStack, for example) and synchronization types (SemaphoreSlim, SpinLock, SpinWait and more).

Visual Studio 2010 brings in support for parallel extensions via a new Parallel Stacks Window, a Parallel Tasks Window and a Concurrency Visualizer. These windows are all designed to give you a better idea of where various tasks are at a given moment in their execution. The Parallel Stacks Window shows how multiple tasks are working their way through their dedicated paths and displays call stacks for the tasks. You have the option to view it in terms of task abstraction or directly as threads. **Figure 4** shows the Parallel Stacks window at runtime displaying the task abstraction view option.

The Parallel Tasks Window has been added specifically to support the new task-based programming model. When your application is in break mode, you can use this window to see things such as the list of tasks, the currently executing method for each task, the affiliated thread, application domain, task ID and more. The window is more than a mere static view; you can click a specific task and the IDE will show its status by bringing the currently executing code to the front. **Figure 5** shows this window for a sample application. Using this window side by side with the Parallel Stacks Window is a nice way to get a quick view into your executing code.

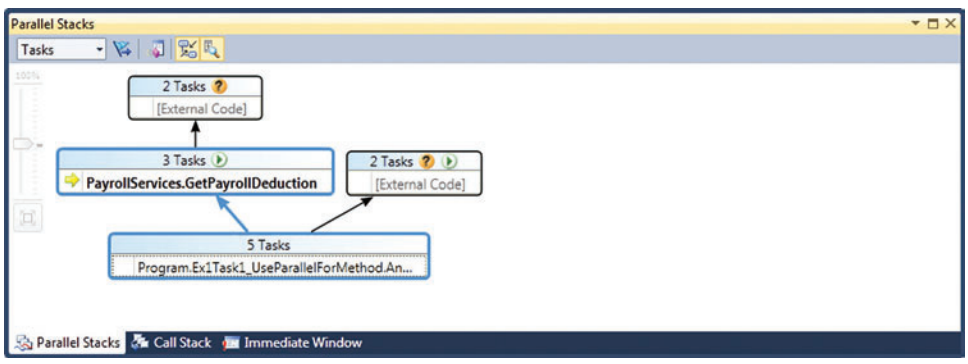


Figure 4 **Parallel Stacks Window in Visual Studio 2010**

Finally, to get a deeper analysis of a parallel application, there is a new Concurrency Visualizer in Visual Studio 2010. This window is much more complex than the other two windows and does not currently support Web projects. Its purpose is to give you some insight as to how your application interacts with its environment in multicore and multithreaded scenarios. There are three views:

- CPU Utilization View shows processor activity.
- Threads View shows how the threads in your application interact.
- Cores View provides specific insight on how your threads migrate across cores.

The Concurrency Visualizer depends on Event Tracing for Windows, which means it needs to run in Windows Vista or later. The views—ranging from basic text through full graphical displays—find inefficiencies in your code caused by poorly distributed load across CPUs, execution bottlenecks, contentions and other potential performance inhibitors. **Figure 6** shows the Concurrency Visualizer displaying thread usage for a profiled application.

With this release, the Microsoft AJAX Library and the AJAX Control Toolkit have been combined.

Web Updates

Along with the general updates I discussed, you'll find some notable improvements for Web developers who move to Visual Studio 2010. Of course, the new Model-View-Controller (MVC) programming model tooling is now included, and I'll cover it in the next section. And there are improved Web development experiences for both

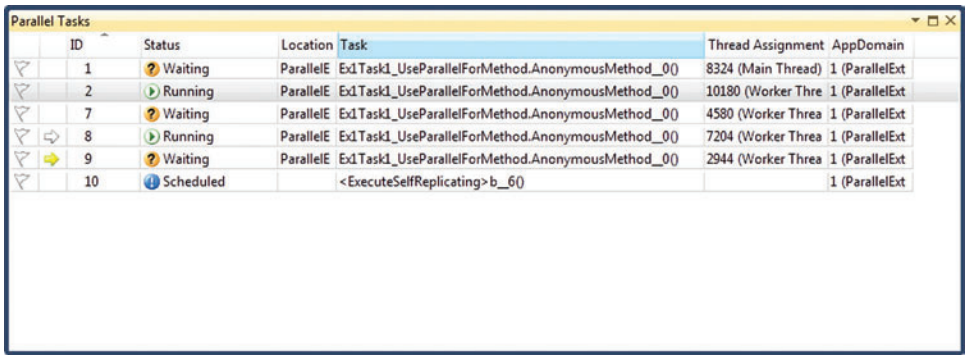


Figure 5 **Parallel Tasks Window in Visual Studio 2010**

client- and server-side Web technologies, as well as a new one-click Web deployment model. But I'll begin with ASP.NET AJAX and the IDE's associated new capabilities.

ASP.NET AJAX tooling existed in Visual Studio as of Visual Studio 2008. However, the server-centric templates have led some developers to believe that it did not add significant value to client-side development. While client-side capabilities have always been there, Visual Studio 2010 includes additional support that really surfaces both the client and server-side capabilities of ASP.NET AJAX. Client-side templates and controls are a significant part of the new features list, as they empower you to take advantage of the improved client data access capabilities, but many other new items such as the jQuery integration are also worth covering.

Prior to Visual Studio 2010 and the .NET Framework 4, if you wanted to use the software download and incorporate the Microsoft AJAX Library with Visual Studio, you got the whole thing injected into your page when you added the ScriptManager. With this release, the Microsoft AJAX Library and the AJAX Control Toolkit have been combined. Also, the Microsoft AJAX team refactored the libraries so you can now ask for the individual pieces you want, rather than be forced into an all-or-nothing choice. You can specify a mode, requesting all, none or specific pieces of the Microsoft AJAX Library.

How It Works *In the ScriptManager tag, simply include the AjaxFrameworkMode attribute. You can set it to Disabled, Enabled or Explicit:*

```
<asp:ScriptManager ... AjaxFrameworkMode="Disabled"/>
```

Disabled means you don't want the Microsoft AJAX Frameworks loaded. Enabled means you want the existing behavior from previous versions (loading the full library). Explicit lets you specify which Microsoft AJAX Library files you want loaded. You can verify this at runtime using View Source with the page loaded.

Similarly, the new Content Delivery Network (CDN) attribute lets you use the most up-to-date versions of script libraries. Previously, you included script libraries such as jQuery or Microsoft AJAX in your project, and the versions you included were the versions you were stuck with for that release. Now, you can request that the latest version be downloaded from Web, rather than stuffing a fixed version into your solutions.

How It Works *In the asp:ScriptManager tag, you can include the EnableCdn attribute, setting it to "true":*

```
<asp:ScriptManager ... EnableCdn="true"/>
```

ASP.NET AJAX 4.0 also lets you define purely client-side templates. You set up a placeholder <ItemTemplate> and let the client do the rendering based on runtime values, such as a JSON-bound object. The runtime does the instantiation on your behalf, and you don't have to do the DOM coding to make it work.

One of the most useful additions to ASP.NET AJAX 4.0 is the improved client data access, which uses a new DataView control and client templates to provide easy access and two-way data binding in the client. It's designed to consume ASMX Web Services, WCF Services, WCF Data Services, MVC Controllers or really anything that returns JSON, and it does this completely from client-side code.

Visual Studio 2010 ships with diagnostic tools to help debug and analyze parallel applications.

The DataView control is the primary control for supporting client templates. It functions similarly to a ListView control, but it's implemented purely in client code. It lets you define a template in the client, pull in the data at runtime and then apply formatting to the records as you display them.

How It Works *You can set the stage with a very basic setup: create an Entity Data Model over the Northwind SQL database and use a vanilla WCF Data Service that pulls it straight through with no changes, meaning the Employees table should be available for query. With that*

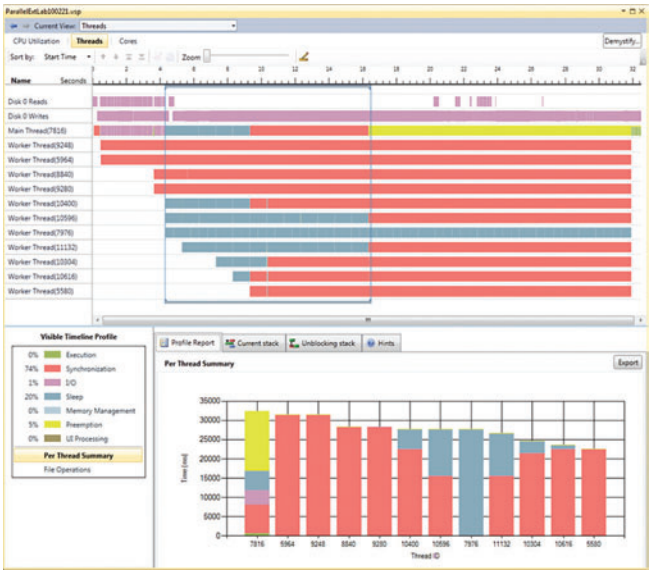


Figure 6 **The Concurrency Visualizer in Visual Studio 2010**

Figure 7 **Improved Data Access with the DataView Control**

```
...
   <script type="text/javascript">
    Sys.require([Sys.components.dataView,
                 Sys.components.adoNetServiceProxy], function () {

       var adoNetDataServiceProxy =
         new Sys.Data.AdoNetServiceProxy('NWDataService.svc');

       Sys.create.dataView("#employeetemplate", {
         dataProvider: adoNetDataServiceProxy,
         fetchOperation: "Employees",
         autoFetch: true
            });
       });
    </script>
</head>
<body>
  <div id="employeetemplate" class="sys-template">
      <span>{{ FirstName }}</span><br/>
  </div>
</body>
```

backdrop, you select the control you want to serve as the template and give it an ID attribute and a class="sys-template" attribute. When you create the data view, the ID is used to signify the destination for the data pulled through on the bind. You can bind manually, but there is also a syntax using curly braces to directly embed the field name in the markup, which will be replaced when the data is fetched and bound to it. There are multiple syntaxes to accomplish this, but one of the more readable ways to write the script and markup to execute this is using the double curly braces, as shown in **Figure 7**.

Beyond the realm of ASP.NET AJAX, there are several other nice enhancements to the ASP.NET programming experience.

One more ASP.NET AJAX item of note is that it has been set up to integrate much more closely with jQuery. With a goal of letting jQuery developers take advantage of the controls in ASP.NET AJAX, all the AJAX Control Toolkit controls are now exposed as jQuery plug-ins automatically. This means jQuery developers can use the controls without having to change their style of development. You can simply think of them as extensions to jQuery.

Beyond the realm of ASP.NET AJAX, there are several other nice enhancements to the ASP.NET programming experience. URL Routing with Web Forms gives URLs a clean, logical feel that is much more search engine optimization (SEO)-friendly, as well as more human-readable. You can define routes in your global.asax file, which maps the requests to the appropriate resources. Not only does this help improve SEO, the pages can take on a predictable path for the user, as shown in **Figure 8**.

There are many more Web-related enhancements worth covering, such as starter Web sites, cleaner HTML and smaller configuration files, but I can't discuss them all here, so I'll move ahead to the MVC tooling additions.

MVC Integration

The MVC architectural pattern is a popular way to build highly maintainable and testable Web applications with well-defined boundaries and clear categorization of code by function. MVC is a style of ASP.NET programming that takes full advantage of the ASP.NET framework. It's an alternative to Web Forms, but it is not a replacement; both models work very well for certain scenarios and skill sets. Developers now have a choice of programming models for Web applications, both based on ASP.NET and fully supported in Visual Studio 2010.

Version 2 of MVC and its affiliated tooling—built between releases of Visual Studio—have been integrated into the Visual Studio 2010 IDE with two project templates, as shown in **Figure 9**.

With the Visual Studio 2010 release, the MVC programming model has focused on three primary areas of improvement. The first is a better separation of concerns. This means more maintainable code through things such as a new RenderAction method for composing business logic separation, and Areas, which let you create "sub-applications" to divide up the functionality in your application. The second area of improvement is a better validation story, made easier through Data Annotations and better bubbling up of validation rules to the client. Finally, there are helper improvements. There are now strongly typed helpers, as well as templated helpers, which let you automatically generate UI for entities.

How It Works *RenderAction makes it easy to share a piece of logic across multiple Controllers. To call the RenderAction method from within your View, use the following script (which calls the NewEmployee action within the HRController):*

```
<%  Html.RenderAction("NewEmployee" "HRController"); %>
```

SharePoint Integration

Another significant integration of tooling into Visual Studio 2010 is the addition of templates for SharePoint Foundation (formerly Windows SharePoint Services). Visual Studio 2010 ships with 12 new templates for SharePoint Foundation, plus the SharePoint 2007 Sequential and State Machine Workflow templates from the Visual Studio Tools for Office add-in to Visual Studio 2008. These templates elevate SharePoint development to first-class status, with a focus on productivity and flexible deployment.

SharePoint Foundation itself is shifting in core usage, moving from being a simple portal you can extend, to a platform for building applications from the ground up, using components such as user management, the underlying list infrastructure and site model. There are significant advances in data access, line-of-business application integration and workflow. Also, there is a more flexible deployment model. Many companies now have massive SharePoint installations and are necessarily more careful about ad-hoc deployment of new SharePoint applications. The new flexible deployment

Figure 8 **URL Routing Improvements in Visual Studio 2010**

New URL Routing Format:

```
/CurrentPosition/Bob/Tester
```

Old URL Style:

```
/CurrentPositions.aspx?name="Bob"&role="Tester"
```

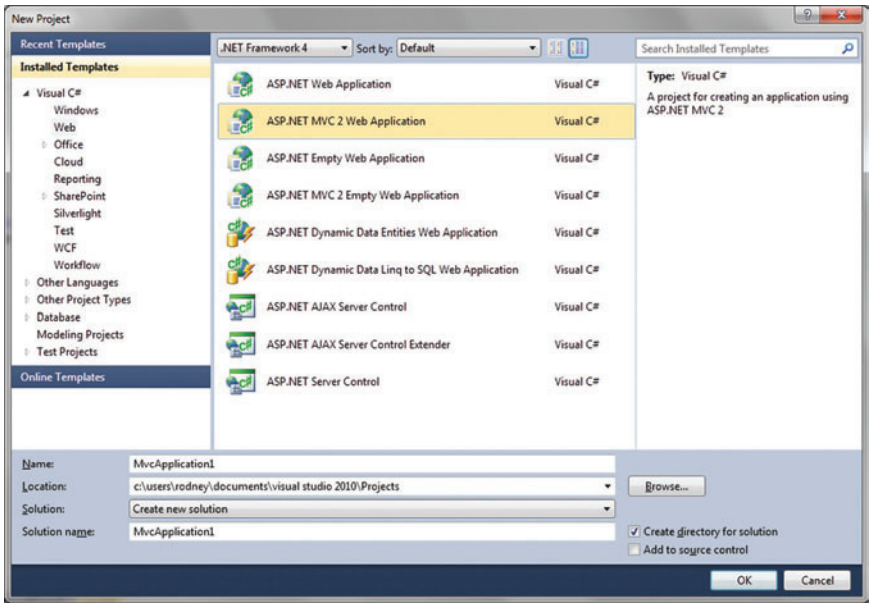


Figure 9 **MVC Project Templates in Visual Studio 2010**

model means you can now sandbox your new installations to specific groups, rather than requiring them to be deployed across the full SharePoint ecosystem.

Visual Studio 2010 has vastly improved the tooling options for building SharePoint applications. True, powerful capabilities for this already existed, but the initial learning curve was fairly tough. With this release, you'll find excellent support for SharePoint built right into the tools. This includes full, integrated debugging, flexible deployment tooling and improved design support for Business Connectivity Services (BCS), Workflow, LINQ to SharePoint and Visual Web Parts. There is also a better bridge for importing items from the SharePoint Designer tool into Visual Studio 2010.

How It Works *Building a visual Web part is considerably easier in Visual Studio 2010 with the new design surface. Begin by choosing to start a new project and selecting SharePoint | Visual Web Part, as shown in* **Figure 10**.

Note that you need to have SharePoint installed on the developer machine for this, but that is not as hard as it used to be. If you are using Windows Vista SP1 (or later) or Windows 7, you can now install SharePoint on that machine directly and not have to worry about installing Windows Server or setting up a VM for it.

You will be prompted to tell Visual Studio 2010 where you want to install the solution for debugging. Once you have the editor open, you can click on the bottom-left Design tab and begin visually constructing your Web part simply by dragging and dropping controls onto the surface. Note that you may need to open the toolbox (View menu | Toolbox) if it is not already displayed.

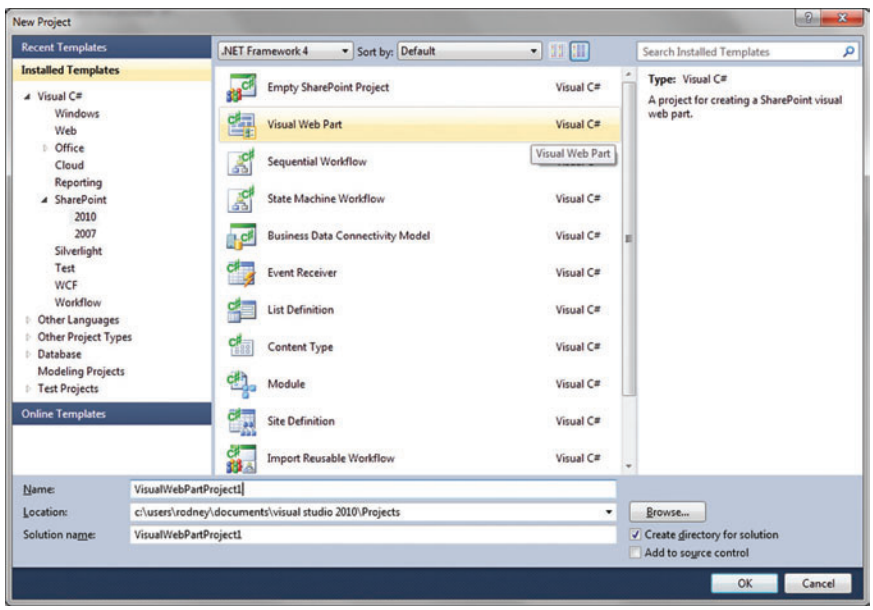


Figure 10 **SharePoint Visual Web Part Template in Visual Studio 2010**

Windows 7 Development

With the growing popularity of Windows 7, developers may want to begin adding functionality to their applications to take advantage of some of the rich experiences available on the new platform. There are numerous ways developers can add Windows 7-specific functionality. One of the popular new capabilities is the new Taskbar. Given its early acceptance among users and ease of programming, it's well positioned for use in applications that target Windows 7.

The Taskbar replaces the Quick Launch taskbar in previous versions of Windows. The Taskbar runs along the bottom (by default) of the Windows 7 screen and allows applications to be "pinned" to it. Moreover, it allows applications to incorporate elements such as jump lists, which are sets of quick links you can get by right-clicking the docked icons, and taskbar previews, which are miniature displays of an application's open windows. The Taskbar is easily programmable, both from managed (via the Windows 7 API Code Pack) and unmanaged code, and should be squarely in the crosshairs of developers who want to take advantage of building applications to run on Windows 7.

How It Works *Users can pin and unpin applications to the Taskbar simply by right-clicking icons, either in the Start Menu, Explorer or the Taskbar itself. To pin an app, right-click the application (not a document, but the application itself) icon and select "Pin to taskbar." To remove it from the Taskbar, do the same thing but select "Unpin this program from Taskbar."*

The Taskbar offers both cosmetic and functional improvements for your applications. Any program can be pinned to the

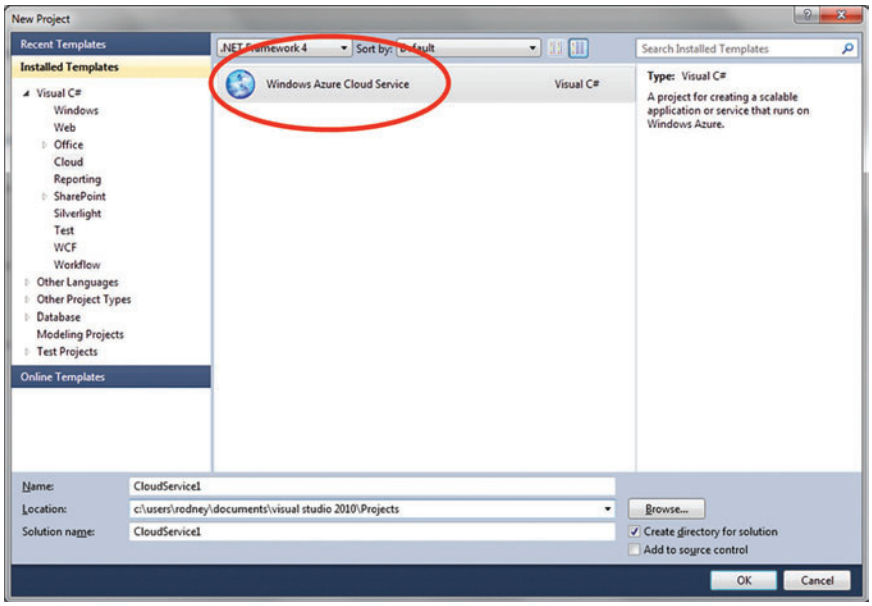


Figure 11 **Selecting a Cloud Project Type in Visual Studio 2010**

Taskbar, whether running or not. The pinned icon can either launch the application or serve to represent minimized windows. When you place the cursor over an icon in the taskbar, it causes a nice visual effect called hot-tracking when the application is running. Hot-tracking takes the dominant icon color and uses it to highlight the icon. Of course, you'll also see a preview of your running instances of the application just above the pinned icon.

The Taskbar also has a feature called aero-peek. To see this, you can place the cursor over a window preview and the window for the selected preview will snap to the front, with the other windows dimmed. This is a great way for users to take a quick peek at a hidden or minimized window without changing focus or pressing a single key.

For tracking apps, instances of running applications are given an application ID and aligned to their application icon in the Taskbar. As a developer, you can use the TaskbarManager and JumpList classes to programmatically interact with the Taskbar from your applications. Note that to use these classes, you will need to download and reference the assemblies in the free Windows 7 API Code Pack from code.msdn.microsoft.com/WindowsAPICodePack

This code pack is built by the Windows SDK team and lets you programmatically work with the Taskbar. It also contains several sample applications that have Windows 7 enhancements. You actually set a reference to the compiled assemblies (in this example, browsing for the Taskbar project) within your code.

You also have the flexibility to give multiple instances of an application either the same application ID or unique application IDs, so you can either group or separate them on the Taskbar. To give different instances of an application unique IDs, you need to set the application ID prior to calling Application.Run. If you merely want to give child windows their own icon in the Taskbar, you can change their application ID at any time.

How It Works *To give a child window its own icon in the Taskbar, use the JumpList class and call the static method CreateJumpListForIndividualWindow, passing in a child window application ID and the window's handle. The code looks like this:*

```
childWindowJumpList = JumpList.CreateJumpListForIndividualWindow(
childWindowApplicationId, this.Handle);
```

Windows Azure Integration

Cloud computing is rapidly emerging as a viable alternative to traditional on-premise and Web hosting architectural solutions. In a nutshell, it's the idea of taking part (or all) of your software solutions and having a company host everything in a massive datacenter, with on-demand scaling and high reliability. The Windows Azure Platform is Microsoft's cloud-computing and services platform. It consists of three sets of services: Windows Azure, SQL Azure and the Windows Azure Platform AppFabric.

Windows Azure is the environment for hosting applications; you can think of it as the OS in the cloud. SQL Azure is essentially the database in the cloud. And the Windows Azure AppFabric is a set of common building blocks used by applications that are hosted in the cloud. The Windows Azure AppFabric currently includes two services: The Access Control Service (for federated authentication and claims-based authorization) and the Service Bus (for connectivity between solutions in the cloud and on-premise solutions).

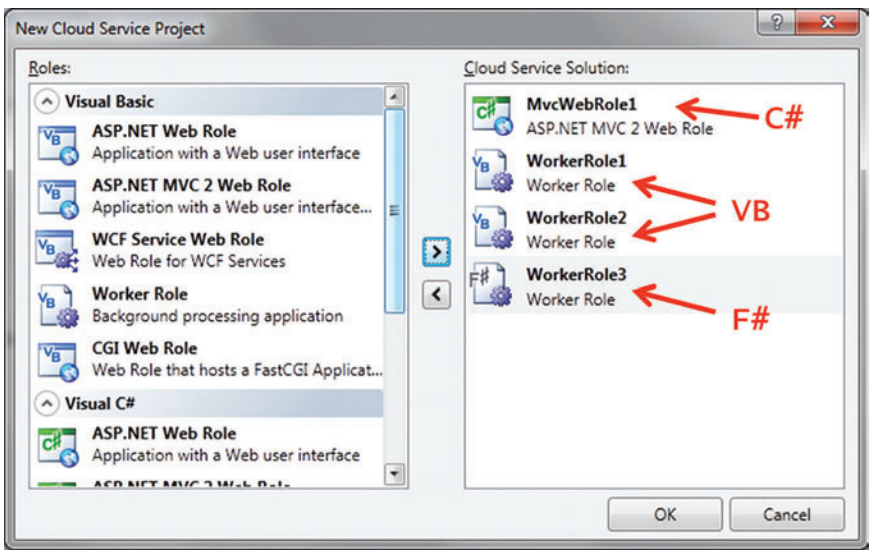


Figure 12 **Setting up a Cloud Service Project with Multiple Roles and Languages**

Visual Studio 2010 includes multiple project templates for building Windows Azure Platform solutions. For both VB.NET and C#, there are four different templates for Web roles and one for a worker role. There is also a worker role project template for F#. The Web role templates for VB.NET and C# include one each for ASP.NET, MVC with ASP.NET, WCF Services and CGI.

One unique aspect of working with Windows Azure Platform projects in Visual Studio 2010 is that you can select multiple roles for your project when you create it. For example, you can create a new project with an ASP.NET MVC Web role in C# and a worker role in VB.NET. This is why the New | Project experience is slightly different for cloud applications; you actually select the roles out of a dialog box rather than via the single selection paradigm you see with most other project templates.

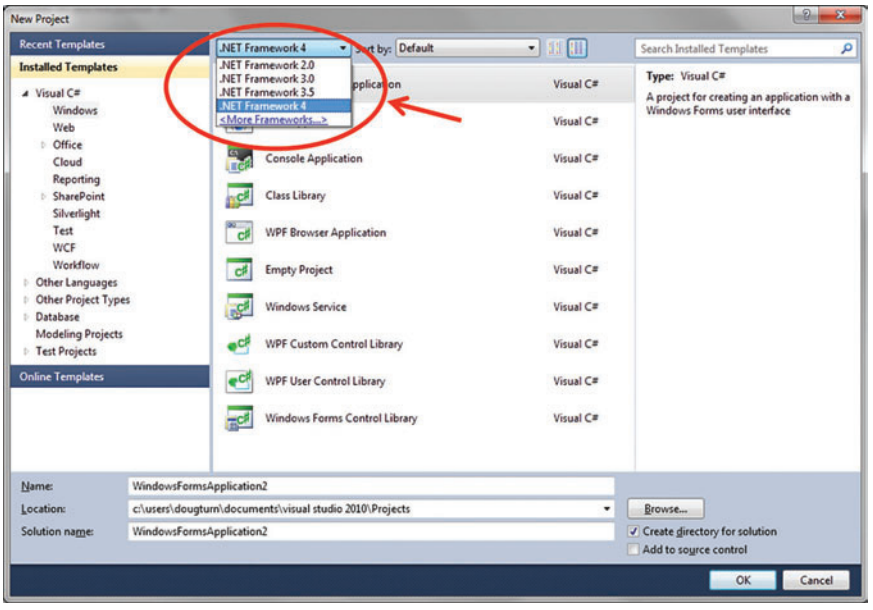


Figure 13 **Selecting a Target Framework Version Via Multi-Targeting**

How It Works *To create a project that targets the Windows Azure Platform in Visual Studio 2010, you select Cloud from the Installed Templates list, as shown in* **Figure 11**.

This will pop up a dialog box where you can select the desired roles, with each role potentially being a different type and/or language. **Figure 12** *shows a new Windows Azure cloud service project with an ASP.NET MVC Web Role, two VB.NET Worker Roles and an F# Worker Role.*

It's somewhat surprising to see how many people don't realize the benefits of multi-targeting and how it can allow them to use Visual Studio 2010 to build solutions that will run on earlier versions of the .NET Framework.

Multi-Targeting

Multi-targeting itself is not actually new to Visual Studio 2010. However, it has a new extensibility feature that's worth a look. It's somewhat surprising to see how many people don't realize the benefits of multi-targeting and how it enables them to use Visual Studio 2010 to build solutions that will run on earlier versions of the .NET Framework. Multi-targeting is there for development teams that want to take advantage of the latest tooling—with its advanced debugging and improved editor features—but are not prepared to move their code (or perhaps customers) to the .NET Framework 4 just yet. With multi-targeting, you have the option of building applications to target the .NET Framework versions 2.0, 3.0, 3.5 or 4.

How It Works *When starting a new project, simply select the targeted framework in the dropdown list as shown in* **Figure 13**.

A new option entitled <More Frameworks...> (note the final entry in the circled drop-down box in **Figure 13**), allows extensibility in the supported target frameworks. This allows the potential addition of future versions of the framework, and perhaps even other .NET Framework profiles further down the road. Such additions would most likely be installed via MSI, but ultimately could be manually added with an XML description file and copying the framework assemblies directly onto the machine.

And More

There are many other new features in Visual Studio 2010, far more than I have been able to cover in this article. It's hard to overlook the customizable start page, which gives great opportunities for companies to tie in company-specific resources. And then there are the C++/ANSI updates, the constantly updated Help, the new F# programming language, Entity Framework enhancements, code visualizations and many other new features worth learning. These capabilities serve to further position Visual Studio 2010 as a compelling development tool, one that is capable of delivering the kind of software our industry now expects. Hopefully, you now have a better idea of what is available in Visual Studio 2010. If you would like to try it out, download an evaluation copy at microsoft.com/vstudio. ■

DOUG TURNURE *is a program manager with the Visual Studio team at Microsoft, primarily focusing on customer feedback and adoption. At different times in his career he has been a developer, author, trainer, tweeter, marketer and occasional conference speaker. Turnure now enjoys living in Seattle, after vowing for many years he would never move there.*

THANKS to the following technical experts for reviewing this article: *Miguel Castro, Mark Dunn and Jim Wooley*

LEADTOOLS®

The World Leader in Imaging Development SDKs

Visit LEAD:
Booth #521
at the Microsoft
Visual Studio Launch
Expo In Las Vegas

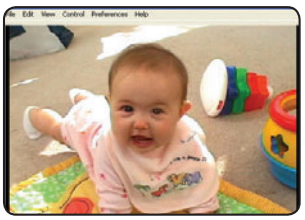
.NET, WPF, WCF, WF, C API, C++ Class Lib, COM & more!

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

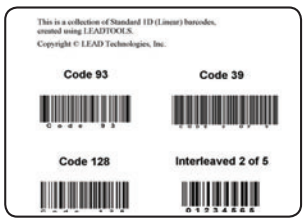
LEADTOOLS provides developers easy access to decades of expertise in color, grayscale, document, medical, vector and multmedia imaging development. Install LEADTOOLS to eliminate months of research and programming time while maintaining high levels of quality, performance and functionality.

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

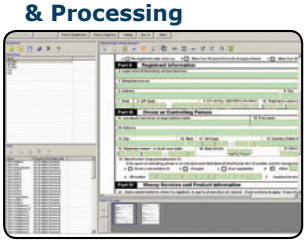
Multimedia



Barcode



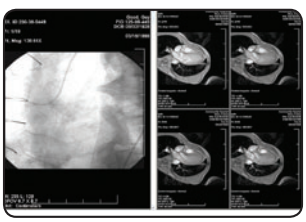
Form Recognition & Processing



Mark-up

Document

DICOM Medical



High Level Design • Low Level Control

LEAD TECHNOLOGIES INCORPORATED

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

What's New in Visual Basic 2010

Jonathan Aneja

From its inception in 1991, the Visual Basic language has always been a phenomenal productivity tool for building applications. Almost 20 years later, it continues to provide easy access to the Microsoft .NET Framework, allowing developers to write applications that span desktops, phones, browsers and even the cloud.

Microsoft this month will ship Visual Studio 2010, which incorporates version 10 of Visual Basic (sometimes referred to as VB 2010 or VB10). This release, the most powerful yet, contains numerous time-saving features that help developers get more done with fewer lines of code. Here's everything you need to know to hit the ground running with Visual Basic in Visual Studio 2010.

This article discusses prerelease versions of Visual Basic 2010 and Visual Studio 2010. All information is subject to change.

This article discusses:

- Coevolution in Visual Basic and C#
- Implicit line continuation
- Statement lambdas
- Auto-implemented properties
- Collection initializers
- Array literals
- Dynamic Language Runtime
- Generic variance

Technologies discussed:

Visual Basic 2010, Visual Studio 2010, Microsoft .NET Framework 4

Coevolution

In the past, Visual Basic and C# were developed by separate teams, which often resulted in features appearing in one language before the other. For example, C# had auto-implemented properties and collection initializers, which weren't in Visual Basic, and Visual Basic had features such as late binding and optional parameters that weren't in C#. But whenever a feature appeared in one of the languages, many customers would ask to have the capability added to the other as well.

To address this feedback, Microsoft merged the Visual Basic and C# teams, embracing a strategy of coevolution. The intent is to make the languages advance together. When major functionality is introduced in one language, it should appear in the other as well. This doesn't mean that every feature will be in both languages and work exactly the same way; indeed, each language has its own history, spirit and feel—traits that are important to maintain. Coevolution does mean that any task you can do in one language should be as simple in the other.

In the .NET Framework 4, both Visual Basic and C# have taken giant strides toward this goal, each adding a number of capabilities the other already had. Coevolution isn't just about the past, though; it's also the strategy for future innovation in the

languages. In that spirit, the .NET Framework 4 introduces powerful new features, such as the Dynamic Language Runtime, Embed Interop Types and generic variance, in both languages simultaneously, allowing Visual Basic and C# developers to take full advantage of the .NET Framework.

New Features in Visual Basic 2010

The new features in Visual Basic 2010 are designed to help you get more done in fewer lines of code. We (the Visual Basic design team) looked at places where developers often have to write a lot of tedious boilerplate code and investigated ways to get the compiler to do the work instead. That's the big picture; now let's delve into some features one by one.

Implicit Line Continuation

Visual Basic is a line-oriented language that uses clear, English-like syntax to enhance readability. But that often results in code that runs up against the 80-character-per-line limit, forcing developers to scroll a lot. You can use the underscore character to tell the compiler that it should keep processing the next line as part of the current one (that is, treat multiple physical lines as a single, logical line). But having to type underscores repeatedly has always been annoying, and in fact, for years the No. 1 feature request has been for the compiler to "just figure it out."

Well, in Visual Basic 2010, the compiler can. It now knows which tokens (such as commas, parentheses and operators) tend to occur right before the line-continuation character, and it inserts the character so developers no longer need to. For example, ending a Visual Basic statement with a comma is never legal; the compiler knows this, so when it sees a token stream that looks like *{comma, enter}*, it infers the presence of the line continuation character, as the example in **Figure 1** shows.

In Visual Basic 2008, the code in **Figure 1** would have needed nine underscores. In each of these cases, though, the compiler inferred when the underscore was necessary and allowed it to be omitted:

- After the <*Extension()*> attribute
- After the ((open paren) in the method declaration
- After the , (comma) for the first parameter
- Before the) (close paren) in the method declaration
- After the = (equal sign)
- After the <%= (opening tag for an embedded expression)

Figure 1 **Inferring Line Continuation**

```
<Extension()>
Function FilterByCountry(
  ByVal customers As IEnumerable(Of Customer),
  ByVal country As String) As IEnumerable(Of Customer)
    Dim query =
      From c In customers
      Where c.Country = country
      Select <Customer>
        <%=
          c.Name &
          "," &
          c.Country
        %>
      </Customer>
    Return query
  End Function
```

- After each & (ampersand) in the XML literal
- Before the %> (closing tag for an embedded expression)

This new compiler capability is especially handy for the method signature, which would go well beyond 80 characters in the example shown if each part were on the same line. In **Figure 2** you'll see all the combinations of tokens and placements in which the line-continuation character is implicit.

As you can see, there are more than 60 places where the language doesn't require underscores. (In fact, none of the code samples in this article required the line-continuation character.) Of course, you can still use the underscore, so code from previous versions of Visual Basic will still compile as expected.

Statement Lambdas

The term lambda can sound intimidating at first, but a lambda is simply a function defined inside another function. Visual Basic 2008 introduced lambda expressions with the Function keyword:

```
Dim customers As Customer() = ...

Array.FindAll(customers, Function(c) c.Country = "Canada")
```

Lambda expressions give you a nice compact way of expressing logic locally without having to split it across multiple methods. For example, here's how the previous code would have looked in Visual Basic 2005 (which didn't support lambda expressions):

```
Dim query = Array.FindAll(customers, AddressOf Filter)

    ...

Function Filter(ByVal c As customer) As Boolean
  Return c.Country = "Canada"
End Function
```

Unfortunately, Visual Basic 2008's lambda expressions required that the expressions return a value, so this:

```
Array.ForEach(customers, Function(c) Console.WriteLine(c.Country))
```

would have caused this:

```
'Compile error: "Expression does not produce a value."
```

Console.WriteLine is a Sub procedure (void, in C#), so it doesn't return a value, which is why the compiler gives an error. To deal with this, Visual Basic 2010 introduces support for statement lambdas, which are lambdas that can contain one or more statements:

```
Array.ForEach(customers, Sub(c) Console.WriteLine(c.Country))
```

Because Console.WriteLine doesn't return a value, we can just create a Sub lambda rather than a Function lambda. Here's another example that uses multiple statements:

```
Array.ForEach(customers, Sub(c)
                           Console.WriteLine("Country Name:")
                           Console.WriteLine(c.Country)
                         End Sub)
```

When this code runs, it'll print two lines for each customer. Also notice that if you hover over *c* when you're coding, you'll see that the compiler has inferred the type as Customer (it's also legal to type *c As Customer* to state the type explicitly). Dynamically wiring up event handlers is another great use for statement lambdas:

```
AddHandler b.Click, Sub(sender As Object, e As EventArgs)
                      MsgBox("Button Clicked")
                      'insert more complex logic here
                    End Sub
```

And, in fact, you can combine statement lambdas with a feature introduced in Visual Basic 2008: relaxed delegates. (You can use delegates—type-safe pointers to functions—to execute multiple

Figure 2 **Where Continuation Characters Are Implicit**

Token	Before	After
, (comma), . (dot), > (attributes), ({ (open brackets), <%= (begin embedded expression (XML literals))		X
), }, ,] (close brackets), %> (close embedded expression)	X	
All LINQ keywords: Aggregate, Distinct, From, Group By, Group Join, Join, Let, Order By, Select, Skip, Skip While, Take, Take While, Where, In, Into, On, Ascending, Descending	X	X
Operators: +, -, *, /, \, ^, >>, <<, Mod, &, +=, -=, *=, /=, \=, ^=, >>=, <<=, &=, <, <=, >, >=, <>, Is, IsNot, Like, And, Or, Xor, AndAlso, OrElse		X
With (in an object initializer)		X

functions at once.) This combination produces an even simpler signature:

```
AddHandler b.Click, Sub()
                      MsgBox("Button Clicked")
                    'insert more complex logic here
                  End Sub
```

Delegate relaxation lets you completely omit the parameters from an event handler—a nice benefit, given that frequently they're not used at all, so they just add visual noise.

In addition to the single-line Sub lambdas and multi-line Sub lambdas we've seen so far, Visual Basic 2010 also supports multi-line Function lambdas:

```
Dim query = customers.Where(Function(c)
                              'Return only customers that have not been saved
                              'insert more complex logic here
                               Return c.ID = -1
                            End Function)
```

Another interesting aspect of statement lambdas is the way they intersect with the anonymous delegates Visual Basic 2008 introduced. People often confuse these with C#'s anonymous methods, though technically they're not the same. Anonymous delegates occur when the Visual Basic compiler infers a delegate type based on the method signature of a lambda:

```
Dim method = Function(product As String)
               If product = "Paper" Then
                 Return 4.5 'units in stock
               Else
                 Return 10 '10 of everything else
               End If
             End Function

MsgBox(method("Paper"))
```

If you run this code, you'll see the value 4.5 displayed in the message box. Also, if you hover over *method*, you'll see the text *Dim method As <Function(String) As Double>*. Because we provided no actual delegate type, the compiler will generate one automatically, like this:

```
Delegate Function $compilerGeneratedName$(product As String) As Double
```

This is called an anonymous delegate, because it appears only in the compiler-produced code, not in the written code. Notice that the compiler inferred the return type as Double, when in fact there was no As clause given to specify the lambda's return type. The compiler looks at all the return statements inside the lambda and sees the types Double (4.5) and Integer (10):

```
'Notice the "As Single"
Dim method = Function(product As String) As Single
               If product = "Paper" Then
                 Return 4.5 'units in stock
               Else
                 Return 10 '10 of everything else
               End If
             End Function
```

It then runs its dominant-type algorithm and determines that it can safely convert 10 to Double but can't safely convert 4.5 to Integer; thus Double is the better pick.

You also can take control of the return type explicitly, in which case the compiler won't attempt to infer the type. Rather than relying on the compiler to infer the delegate type, it's very common to assign a lambda to a variable that has an explicit delegate type:

```
Dim method As Func(Of String, Single) =
  Function(product)
    If product = "Paper" Then
      Return 4.5 'units in stock
    Else
      Return 10 '10 of everything else
    End If
  End Function
```

Because an explicit target type was provided, there's no need to say *As String* or *As Single;* the compiler can infer their presence based on the delegate type from the left-hand side of the statement. Thus, if you hover over *product* you'll find that the inferred type is String. Specifying *As Single* is no longer necessary, because the delegate type already provides that information. In the previous example, the signature of the Func delegate (which the .NET Framework includes) looks like this:

```
Delegate Function Func(Of T, R)(ByVal param As T) As R
```

with one minor exception, as we'll see later in the Generic Variance section.

> For years, the No. 1 feature request has been for the compiler to "just figure it out."

Auto-Implemented Properties

In Visual Basic, properties are class members you use to expose an object's state to the outside world. A typical property declaration looks something like this:

```
Private _Country As String

Property Country As String
  Get
    Return _Country
  End Get
  Set(ByVal value As String)
    _Country = value
  End Set
End Property
```

That's 10 lines of code for what's actually a very simple concept. Given that typical objects often have dozens of properties, you end up including a lot of boilerplate code in class definitions. To make

such tasks easier, Visual Basic 2010 introduces auto-implemented properties, which allow you to define a simple property using only one line of code:

```
Property Country As String
```

In this case, the compiler will go ahead and generate the Getter, Setter and backing fields automatically. The name of the backing field will always be an underscore followed by the name of the property: *_Country* in this case. This naming convention ensures binary serialization compatibility should an auto-implemented property be changed to a regular one. As long as the name of the backing field is the same, binary serialization will continue to work.

One of the cool things you can do with auto-implemented properties is specify initializers that set the property's default value when the constructor runs. A common scenario with entity classes, for example, sets the primary key to something like -1 to indicate that it's in an unsaved state. Here's what that code would look like:

```
Property ID As Integer = -1
```

When the constructor runs, the backing field (_ID) will be set to the value -1 automatically. The initializer syntax also works for reference types:

```
Property OrderList As List(Of Order) = New List(Of Order)
```

The previous line of code may not feel very "Visual Basic-ish," given that entering the name of the type twice is redundant. The good news is there's an even shorter syntax that's consistent with what Visual Basic allows in regular variable declarations:

```
Property OrderList As New List(Of Order)
```

You can even combine this with Object Initializers to allow setting additional properties:

```
Property OrderList As New List(Of Order) With {.Capacity = 100}
```

Obviously, for more complex properties, the expanded syntax is still necessary. You can still type *Property{Tab}* to activate the old property snippet. Alternatively, after typing the first line of the property, you can just enter *Get{Enter}*, and the IDE will generate the old-style property:

```
Property Name As String
  Get

  End Get
  Set(ByVal value As String)

  End Set
End Property
```

Figure 3 **Creating a Custom Dynamic Object and Calling It with Visual Basic Late Binding**

```
Imports System.Dynamic
Module Module1
  Sub Main()
    Dim p As Object = New PropertyBag
      p.One = 1
      p.Two = 2
      p.Three = 3
    Console.WriteLine(p.One)
    Console.WriteLine(p.Two)
    Console.WriteLine(p.Three)
  End Sub
    Class PropertyBag : Inherits DynamicObject
      Private values As New Dictionary(Of String, Integer)
      Public Overrides Function TrySetMember(
        ByVal binder As SetMemberBinder,
        ByVal value As Object) As Boolean
          values(binder.Name) = value
        Return True
      End Function
      Public Overrides Function TryGetMember(
        ByVal binder As GetMemberBinder,
        ByRef result As Object) As Boolean
        Return values.TryGetValue(binder.Name, result)
      End Function
    End Class
End Module
```

> Delegate relaxation lets you completely omit the parameters from an event-handler—a nice benefit.

People often remark that the new property syntax is almost identical to the syntax for a public field, so why not use a public field instead? Well, for a few reasons:

- Much of the .NET data-binding infrastructure works against properties but not fields.
- An interface can't enforce the existence of a field; it can enforce that of a property.
- Properties provide more long-term flexibility for changing business rules. For example, suppose someone introduces the rule that a phone number must be 10 digits. There's no way to perform this validation when assigning to a public field. Changing a public field to a property is a breaking change for scenarios such as binary serialization and reflection.

Collection Initializers

A common .NET practice is to instantiate a collection and then populate it by calling the Add method once for each element:

```
Dim digits As New List(Of Integer)
digits.Add(0)
digits.Add(1)
digits.Add(2)
digits.Add(3)
digits.Add(4)
digits.Add(5)
digits.Add(6)
digits.Add(7)
digits.Add(8)
digits.Add(9)
```

But the result is a lot of syntactic overhead for what's fundamentally a very simple concept. Visual Basic 2010 introduces *collection initializers* to let you more easily instantiate collections. With this code:

```
Dim digits = New List(Of Integer) From {1, 2, 3, 4, 5, 6, 7, 8, 9, 0}
```

the compiler will generate all the calls to the Add method automatically. You can also use the feature with Visual Basic's As New syntax:

```
Dim digits As New List(Of Integer) From {1, 2, 3, 4, 5, 6, 7, 8, 9, 0}
```

Note that on the Visual Basic Team, we always recommend using the second syntax (As New) over the former, because it makes code resilient against changes to the Option Infer setting.

You can use collection initializers against any type that meets the following requirements:

- You can iterate over it using a For Each statement—that is, it implements IEnumerable. (For a more precise/detailed definition of a collection type, see section 10.9.3 of the Visual Basic Language Specification at msdn.microsoft.com/library/aa711986(VS.71).aspx).
- It has an accessible (not necessarily public) parameter-less constructor.
- It has an accessible (not necessarily public) instance or extension method named Add.

That means you can also use collection initializers with more complex types, such as dictionaries:

```
Dim lookupTable As New Dictionary(Of Integer, String) From
  {{1, "One"},
   {2, "Two"},
   {3, "Three"},
   {4, "Four"}}
```

(Note that even though this statement spans five lines, there are no underscores.) In this case, the compiler will generate code that's equivalent to the old way of initializing the dictionary:

```
Dim lookupTable As New Dictionary(Of Integer, String)
lookupTable.Add(1, "One")
lookupTable.Add(2, "Two")
lookupTable.Add(3, "Three")
lookupTable.Add(4, "Four")
```

The compiler is calling an Add method that has *two* parameters instead of one. It knows to do this because the values passed into the collection initializer were in nested braces, like this: {{1, "One"}, {2, "Two"}, …}. For each set of nested braces, the compiler attempts to pass those parameters to a compatible Add method.

You can also provide your own custom Add implementation by using an extension method:

```
<Extension()>
  Sub Add(ByVal source As IList(Of Customer),
          ByVal id As Integer,
          ByVal name As String,
          ByVal city As String)

      source.Add(New Customer With
                 {
                    .ID = id,
                    .Name = name,
                    .City = city
                 })
  End Sub
```

(Look at all those missing underscores!) This method extends any type that implements IList(Of Customer) and then allows you to use the new collection initializer syntax like this:

```
Dim list = New List(Of Customer) From
           {
             {1, "Jon", "Redmond"},
             {2, "Bob", "Seattle"},
             {3, "Sally", "Toronto"}
           }
```

(adding three customers to *list*). You can also use collection initializers in conjunction with auto-implemented properties:

```
Property States As New List(Of String) From {"AL", "AK", "AR", "AZ", ...}
```

Array Literals

In addition to more powerful ways of working with collection types, Visual Basic 2010 also provides some great enhancements for working with arrays. Consider the following code (which works fine in older versions):

```
Dim numbers As Integer() = New Integer() {1, 2, 3, 4, 5}
```

It's obvious from looking at the elements in the array that each is an Integer, so having to actually type out Integer twice in this line doesn't really add any value. *Array literals* allow creation of an array by putting all of its elements inside braces, and then having the compiler infer the type automatically:

```
Dim numbers = {1, 2, 3, 4, 5}
```

The type of *numbers* isn't Object, but rather Integer() (as long as Option Infer is on), because the array literal can now stand by itself and has its own type. Consider a more complicated example:

```
Dim numbers = {1, 2, 3, 4, 5.555}
```

In this case, the type of *numbers* will be inferred as Double(). The compiler determines the type by examining each element in the array and calculating the dominant type (using the same algorithm discussed earlier for inferring the return type of a statement lambda). What happens if there's no dominant type, such as in the following code:

```
Dim numbers = {1, 2, 3, 4, "5"}
```

In this case, converting an Integer to a String would be a narrowing conversion (that is, there would be potential for data loss at runtime), and likewise, converting a String to an Integer would also be a narrowing conversion. The only safe type to pick is Object() (and the compiler will give an error when Option Strict is on).

Array literals can be nested to form either multi-dimensional arrays or jagged arrays:

```
'2-dimensional array
Dim matrix = {{1, 0}, {0, 1}}

'jagged array - the parentheses force evaluation of the inner array first
Dim jagged = { ({1, 0}), ({0, 1}) }
```

Dynamic Language Runtime

While technically a static language at heart, Visual Basic has always had extremely powerful dynamic capabilities, such as late binding. Visual Studio 2010 ships with a new platform called the Dynamic Language Runtime (DLR), which makes it easier to build—and communicate among—dynamic languages. Visual Basic 2010 has been

Figure 4 **An Example of Generic Variance**

```
Option Strict On
Public Class Form1
  Sub Form1_Load() Handles MyBase.Load
    Dim buttons As New List(Of Button) From
      {
        New Button With
        {
          .Name = "btnOk",
          .Enabled = True
        },
        New Button With
        {
          .Name = "btnCancel",
          .Enabled = False
        }
      }

    Dim enabledOnly = FilterEnabledOnly(buttons)
  End Sub
  Function FilterEnabledOnly(
    ByVal controls As IEnumerable(Of Control)
    ) As IEnumerable(Of Control)
    Return From c In controls
    Where c.Enabled = True
  End Function
End Class
```

updated to fully support the DLR in its latebinder, letting developers use libraries and frameworks developed in other languages such as IronPython/IronRuby.

The cool thing about this feature is that nothing's changed syntactically (in fact, not a single line of code was modified in the compiler to support this feature). Developers can still make operations late-bound the same way they did in previous versions of Visual Basic. What has changed is code in the Visual Basic Runtime (Microsoft.VisualBasic.dll), which now recognizes the IDynamicMetaObjectProvider interface that the DLR provides. If an object implements this interface, the Visual Basic Runtime will construct a DLR CallSite and allow the object and its providing language to inject their own semantics into the operation.

For example, the Python Standard Libraries contain a file called random.py with a method called shuffle that can be used to randomly rearrange the elements in an array. Calling it is simple:

```
Dim python As ScriptRuntime = Python.CreateRuntime()
Dim random As Object = python.UseFile("random.py")

Dim items = {1, 2, 3, 4, 5, 6, 7}
random.shuffle(items)
```

At runtime, Visual Basic sees that the object implements IDynamicMetaObjectProvider and thus passes control to the DLR, which then communicates with Python and executes the method (passing along the array that was defined in Visual Basic as an argument to the method).

That's an example of invoking a DLR-enabled API, but it's also possible for developers to create their own APIs that use this feature. The key is to implement the IDynamicMetaObjectProvider interface, in which case the Visual Basic and C# compilers will recognize that the object has special dynamic semantics. Rather than implementing the interface manually, it's easier to inherit from the System.Dynamic.DynamicObject class (which already implements this interface) and just override a couple of methods. **Figure 3** shows a full example of creating a custom dynamic object (a "property bag" that appears to create properties on the fly) and calling it using normal Visual Basic late binding. (For more information on working with DynamicObject, check out Doug Rothaus' excellent article at blogs.msdn.com/vbteam/archive/2010/01/20/fun-with-dynamic-objects-doug-rothaus.aspx.)

Generic Variance

This is a feature that can sound really complicated (with terms like covariance and contravariance) at first, but it's actually pretty simple. If you have an object of type IEnumerable(Of Apple) and want to assign it to an IEnumerable(Of Fruit), that should be legal, because every Apple is a Fruit (enforced by an inheritance relationship). Unfortunately, before Visual Basic 2010, generic variance was not supported in the compiler, even though it actually was supported in the Common Language Runtime (CLR).

Let's consider the example in **Figure 4**.In Visual Basic 2008, the code in **Figure 4** would generate a compile error (or if Option Strict is off, a runtime exception) on the Dim enabledOnly line. The workaround was to call the .Cast extension method, as shown here:

```
'Old way, the call to Cast(Of Control) is no longer necessary in VB 2010
Dim enabledOnly = FilterEnabledOnly(buttons.Cast(Of Control))
```

This is no longer necessary, because in Visual Basic 2010, the IEnumerable interface has been marked as covariant by using the Out modifier:

```
Interface IEnumerable(Of Out T)
  ...
End Interface
```

This means the generic parameter T is now variant (that is, it works for inheritance relationships) and the compiler will ensure it's only used in positions where the type is coming out of the interface. Generic parameters can also be contravariant, which means they're only used in *input* positions. A type can actually have both. For example, the Func delegate discussed earlier has both contravariant parameters (things that get passed in) and a covariant parameter (for the return type):

```
Delegate Function Func(Of In T, Out R)(ByVal param As T) As R
```

> The coolest thing about all the features in Visual Basic 2010 is that you can even use them in projects that target the .NET Framework 2.0 through the .NET Framework 3.5.

You can use the In and Out modifiers on your custom interfaces and delegates. Many commonly used interfaces and delegates in the .NET Framework 4 have already been marked as variant; common examples include all the Action/Func delegates, IEnumerable(Of T), IComparer(Of T) and IQueryable(Of T), among others.

The cool thing about generic variance is that it's a feature you don't really need to worry about—if it's doing its job, you'll never notice it. Situations that used to cause compile errors or require a call to .Cast(Of T) should work just fine in Visual Basic 2010.

Improved Optional Parameters

Optional parameters provide a handy productivity feature that lets developers make more flexible methods and avoid polluting a class with numerous overloads of a method. One limitation in the past was that optional parameters could not be nullable (or indeed any non-intrinsic structure type). Visual Basic 2010 now lets you define optional parameters of *any* value type:

```
Sub DisplayOrder(ByVal customer As Customer,
                 ByVal orderID As Integer,
                 Optional ByVal units As Integer? = 0,
                 Optional ByVal backgroundColor As Color = Nothing)
End Sub
```

In this case, units is of type Nullable(Of Integer) and backgroundColor is a non-intrinsic structure type, but they can still be used as optional parameters. Visual Basic 2010 also provides better support for optional parameters that are generic.

INSPIRE! EMPOWER! IMPRESS!

You've got the data, but time, budget and staff constraints can make it hard to present that valuable information in a way that will impress. With Infragistics' **NetAdvantage for Silverlight Data Visualization**, you can create Web-based data visualizations and dashboard-driven applications on Microsoft Silverlight (and coming soon for WPF) that will not only impress decision makers, it actually empowers them. Go to infragistics.com/sldv today and get inspired to create killer apps.

Infragistics®
KILLER APPS. NO EXCUSES.

Infragistics Sales 800 231 8588
Infragistics Europe Sales +44 (0) 800 298 9055
Infragistics India +91-80-6785-1111

Embed Interop Types

For applications that perform COM Interop, a common pain point is having to work with Primary Interop Assemblies (PIAs). A PIA is a .NET assembly that serves as a Runtime Callable Wrapper (RCW) over a COM component and has a unique GUID to identify it. .NET assemblies communicate with a PIA, which then performs any necessary marshalling to move data between COM and .NET.

Unfortunately, PIAs can complicate deployment because they're additional DLLs that need to be deployed to the end users' machines. They also cause problems for versioning—for example, if you want an application to be able to work against both Excel 2003 and Excel 2007, you'd need to deploy both PIAs with the application.

The Embed Interop Types feature embeds directly into the application, but only the types and members from the PIA that are absolutely necessary, thus removing the need for PIAs to be deployed to the end users' machines.

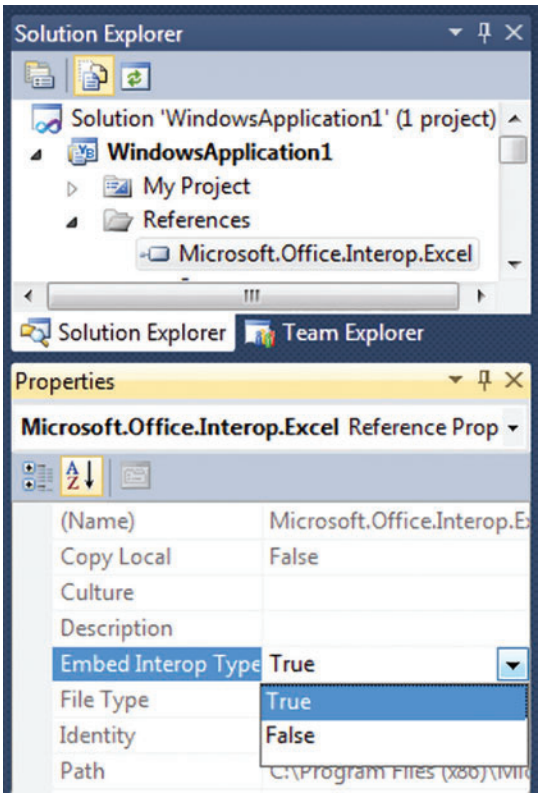


Figure 5 **Enabling the Embed Interop Types Feature in Solution Explorer**

To turn on this feature for an existing project (it's already on by default for new references), select the reference in Solution Explorer and change the Embed Interop Types option in the properties window (see **Figure 5**). Or, if compiling using the command-line compiler, use the /l (or /link) switch instead of the /r and /reference.

Once you've turned this feature on, the application no longer has a dependency on the PIA. In fact, if you open the assembly in Reflector or ildasm, you'll notice that there actually isn't any reference to the PIA at all.

Multi-Targeting

The coolest thing about all the features in Visual Basic 2010 is that you can even use them in projects that target the .NET Framework 2.0 through the .NET Framework 3.5. This means that implicit line continuation, array literals, collection initializers, statement lambdas, auto-implemented properties and so on will all work in existing projects without having to retarget to the .NET Framework 4.

The one exception is Embed Interop Types, which has a dependency on types that are only in the .NET Framework 4; as a result you can't use it when targeting .NET Framework versions 2.0 through 3.5. Also, the types that are marked as variant are only marked that way in the .NET Framework 4, so in the earlier example, you'd still have to call .Cast(Of T) when targeting versions 2.0 through 3.5. You can, however, make your own variant types (using the In/Out modifiers) when targeting those earlier versions.

To change the current target framework for an application, double-click My Project, click the Compile tab, click Advanced Compile Options and then select from the combo box at the bottom.

When compiling from the command line, there's actually no command-line switch to enable this feature. Instead, the compiler looks at which assembly provided the definition of System.Object (typically mscorlib) and which framework that assembly is targeting, then stamps that value in your output assembly. (This is the same mechanism the compiler uses when building Silverlight assemblies.) When using the IDE, all this happens transparently, so in general it's not something you need to worry about.

Try It Out

As you can see, Visual Basic 2010 has many powerful features that let you be more productive while writing fewer lines of code, offloading more work to the compiler. In this article, I've only looked at language features, but the Visual Basic 2010 IDE also has a ton of great enhancements. Here's a partial list:

- Navigate To
- Highlight References
- Generate From Usage
- Better IntelliSense (substring matching, camel-case lookup, suggestion mode—useful for "test first" styles of development)
- Multi-Monitor Support
- Zooming

The Visual Basic team would love to hear your feedback on what we can do to make Visual Basic even better, so send us your comments and questions on Microsoft Connect. To learn more about the language and IDE features, check out the content on msdn.com/vbasic, including the articles, samples and How-Do-I videos. Of course, the best way to learn is by diving in and using the product, so it's time to install it and try it out.

Want more Visual Basic? You got it. MSDN Magazine *is resuming monthly publication of the Basic Instincts column, which focuses on the Visual Basic developer and related topics and is written by the Visual Basic team at Microsoft.* ■

Jonathan Aneja *is a program manager on the Entity Framework team at Microsoft. Previously he was the program manager for the Visual Basic compiler during the releases of Visual Basic 2008 and Visual Basic 2010. He has been at Microsoft for four years.*

Thanks to the following technical expert for reviewing this article: *Dustin Campbell, Jason Malinowski and Lucian Wischik*

ESRI® Developer Network

Integrate Mapping and GIS into Your Applications

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

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

Subscribe to EDN and leverage the power of GIS to get more from your data. Visit www.esri.com/edn.

ESRI

RadControls *for* Silverlight

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

Pioneering Support for Microsoft Silverlight 4

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

Engineered for Great Performance

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

Support for Visual Studio 2010 and Expression Blend

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

A Comprehensive Silverlight Toolset from the Masters of Web UI

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

Code Re-Use with RadControls for WPF

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

US Sales: +1.888.365.2779 • Germany Sales: +49.89.8780687.70 • Europe HQ: +359.2.80.99.850 • e-mail: sales@telerik.com

UI COMPONENTS	DATA	PRODUCTIVITY TOOLS	AUTOMATED TESTING	TFS TOOLS	CMS
ASP.NET AJAX	OpenAccess ORM	JustCode	Web Testing Tools	Work Item Manager	Sitefinity
Silverlight	Reporting			Project Dashboard	
ASP.NET MVC					
WinForms					
WPF					

www.telerik.com

telerik

deliver more than expected

Debugging Applications with IntelliTrace

Justin Marks

How does anyone fix a bug in their code? You set a few breakpoints, run the program under the debugger, do a little single-stepping—and pray the problem falls into your lap so you can get on to other things.

We've been doing the same style of debugging almost since the ENIAC was invented. This tedious and time-intensive debugging approach has served us well, but it's time debugging got easier. With the release of Visual Studio 2010 Ultimate, the new IntelliTrace feature brings debugging into the 21st century by giving developers better insight into their applications' execution.

This article is based on a prerelease version of Visual Studio 2010 Ultimate. All information is subject to change.

This article discusses:

- Using IntelliTrace to track debugging events
- Adjusting the amount of data collected by IntelliTrace
- Investigating a user code error
- Collecting deep call information for application diagnostics
- Eliminating the "no repro" scenario

Technologies discussed:

Visual Studio 2010 Ultimate, IntelliTrace

Code download available at:

code.msdn.microsoft.com/mag201004Debug

Much like other monitoring and tracing tools such as Process Monitor from Windows Sysinternals, Visual Studio 2010 collects data about an application while it's executing to help developers diagnose errors. The collected data is referred to as IntelliTrace events. These events are collected as part of the default debugging experience, and among other things, they let developers step back in time to see what happened in an application without having to restart the debugger.

In this article, I'll introduce you to IntelliTrace and show how it provides value to developers as part of their day-to-day development activities. I'll show how IntelliTrace provides a time line of events that occurred during the execution of an application, and how developers can use these events to aid debugging. Next, I'll discuss the settings developers can change to collect a deeper set of information about the application to get a complete execution history. Finally, I'll show how to use a previously recorded IntelliTrace file that was created by someone else—a tester—to debug an application, without having to run the application to reproduce the error.

When the Visual Studio diagnostics team started planning for Visual Studio 2010, we spent a lot of time talking to customers about how they diagnose problems in their applications. Though everyone has a different pattern and favorite set of tools, one point was overwhelmingly clear: the traditional methods of diagnosing application issues are difficult, time-consuming and costly. The bug reports that developers receive almost never have any steps to reproduce the problem, and mostly consist of statements like "I was using the program and it crashed." In the rare case when decent repro steps

are available, you may be faced with bugs occurring in specific environments, and this leads to a whole new set of problems to solve. Moreover, bugs often are caused by a misunderstanding of how a framework or other code operates.

With these pain points in mind, we set about to create a new debugger feature where the right information was collected at the time the problem occurred. Our goal was to hand developers the exact repro steps and system environment settings, as well as expose the behavior of frameworks and code they use, to drastically improve their diagnosability. With the release of Visual Studio 2010 Ultimate, IntelliTrace brings a greatly improved debugging experience by giving developers better insight into the application and framework behavior, as well as the ability to open an IntelliTrace file collected by a tester to resolve "no repro" scenarios.

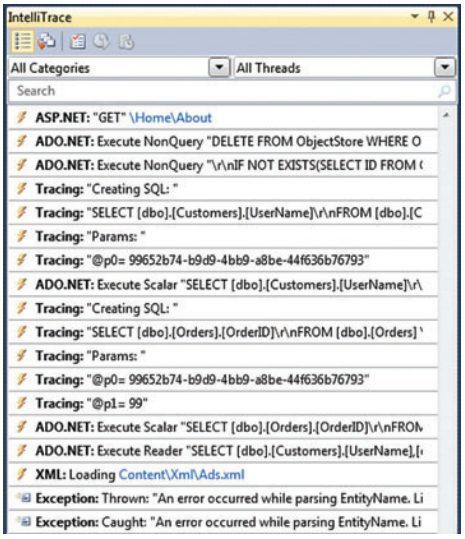


Figure 1 **Diagnostic Information Collected by IntelliTrace**

Introducing IntelliTrace

When a developer needs a deeper understanding of code execution, IntelliTrace offers a way to "turn up the dial" to collect the complete execution history of an application.

To illustrate this, I will use the Tailspin Toys demo application to show you the type of information IntelliTrace can collect. To begin, I'll open up the solution in Visual Studio and start debugging. When the Web site launches, I'll navigate to the "About us" page and receive an error from the server. How might you diagnose the problem? If you're anything like me, your first instinct is to configure the web.config file to not show custom errors and then restart the debugger. But what if this issue is intermittent? Wouldn't it be nice if you could just break into the process at this point, after the error occurred, and get a history of what happened in the application from Visual Studio?

While you're debugging, IntelliTrace collects data about a managed application in the background, including information from many framework components such as ADO.NET, ASP.NET and System.XML. These IntelliTrace events allow the developer to see what has previously occurred during execution, and most importantly, to "step back in time" to see prior states of the application without having to restart the debugger. When I break into the debugger, I'm immediately presented with a sequential list of the IntelliTrace events that were collected (see **Figure 1**).

As you can see from **Figure 1**, the list of IntelliTrace events goes far beyond file and registry accesses as you'd see in Process Monitor. We have defined almost 150 IntelliTrace events for Visual Studio 2010, and we plan on augmenting this list with additional events over time. **Figure 2** highlights some of the categories of events that are collected by IntelliTrace.

The IntelliTrace window lets me filter the list of collected events by category—the categories shown in **Figure 2**—or by thread. In addition, I can do text-based searches to find key events I can quickly jump to. Because IntelliTrace also collects exceptions, I can do a search for the term "exception" and the list will filter to show me the exception that caused the ASP.NET error page, both where it was thrown and where it was caught. In this case, the error was caused by an XMLException while parsing an entity on line 10, position 53 (see **Figure 3**). When I click on the thrown exception event, other debugger windows, such as the Call Stack and Watch windows, show data relative to the event itself, so it's like you were debugging the instant that exception was thrown. In addition, just like walking up a call stack, the editor will open the appropriate source file and highlight the line of code corresponding to the event in orange to represent IntelliTrace.

IntelliTrace has given me a very helpful piece of information for diagnosing the issue: an XML file was loaded and a specific character in the XML was unexpected or incorrect. But I still don't know which file was accessed. Once again, IntelliTrace has collected the information I need—namely the file access.

Figure 2 **IntelliTrace Events Are Available Across the Microsoft .NET Framework**

Category	Description and Collected Data
ADO.NET	Events around executing queries against SQL, the executed command as well as the connection string.
ASP.NET	Events around the ASP.NET pipeline as well as request processing and redirection.
Console	Console output.
Data Binding	Windows Forms data binding.
Environment Variables	Evaluation and retrieval of environment variables from the process.
File	Creation, deletion and access of files.
Gestures	User actions performed against common controls from Web forms, Windows Forms and WPF. In addition to collecting data about the interaction with the control, clicking on one of these events automatically redirects you to the appropriate event handler.
Lazy Initialization	Initialization of lazily loaded variables.
Registry	Creation, deletion and querying of registry information.
Service Model	Web service calls from WCF.
Threading	Queuing of user work items and parallel computing tasks.
Tracing	Debugger trace output and assertions.
User Prompt	Display of Windows Forms and WPF message boxes as well as the result of the dialog.
Workflow	Activity instantiation and completion.
XML	XML file loading.

Looking again at the IntelliTrace window in **Figure 3**, I can see that the event just prior to the exception is an XML file load event for "Content\Xml\Ads.xml." This must be the file causing the error. I can easily open this file in Visual Studio. Looking at line 10, position 53, I see that there is indeed an error in this file, namely that "&b=1" is invalid for the NavigateUrl XML element. By removing these invalid characters, the Web site should now load correctly.

Now I want you to think about the last unhandled exception you debugged with traditional debugging techniques. If it was an exception like this one, you would have seen where the exception occurred, but definitely not the exact reason or the invalid character. This is the key to IntelliTrace—it gives you better information to diagnose issues quicker and easier. You have more important things to do than waste your time hunting around for information.

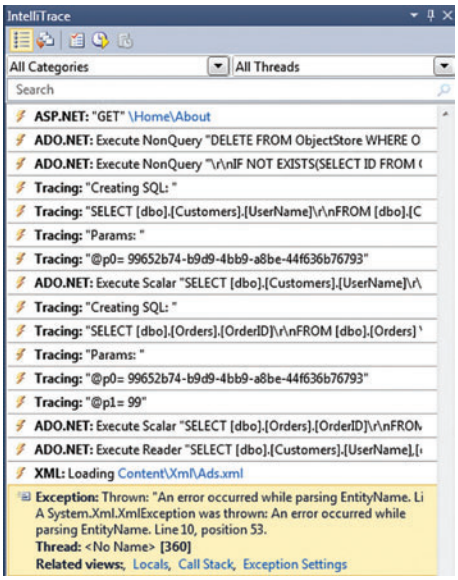


Figure 3 **An XMLException Was Thrown While Parsing an Entity on Line 10, Position 53**

Using IntelliTrace to Track Debugger Events

I've just shown you how IntelliTrace collects exceptions, both handled and unhandled, that occur during an application's execution, and how IntelliTrace events across the framework can provide insight into what an application did under the hood. But that's not all. IntelliTrace also collects events caused by the debugger, namely breakpoints, tracepoints and stepping events.

One of the most common debugging techniques is to set a breakpoint near the point where you think a problem exists and then step through the code watching variables change. This is especially useful when debugging through loops, watching for a variable to change to a specific value. Unfortunately, most developers are impatient and pound the F10 key to step quickly through the code, only to find that they stepped too far. Then they need to restart their debugging session and try again. With IntelliTrace, all breakpoints and stepping events are recorded, along with their contextual data, so you can quickly navigate to points where you stopped before.

If I click on one of these debugger events, the Watch window will show me all the data I previously looked at, including values I evaluated in the Locals, Watch and Autos windows, as well as QuickWatch and DataTips.

Often, previously developed and deployed code doesn't have the needed tracing built in to help debug issues that may arise. Breakpoints offer the ability to see more of what's going on under the hood of the application. But most of the time, the developer doesn't need to stop at the breakpoint; rather he wants to collect some data and continue execution. This is especially the case inside loops where you want a record of the iterator value without having to stop at each iteration. In these scenarios, tracepoints are a great alternative. Tracepoints let the developer make the debugger perform a custom action; that is, execute a macro or print out a trace message, instead of breaking execution. With IntelliTrace, the tracepoints output is collected and can be viewed in the same interface as the other IntelliTrace events (see **Figure 4**).

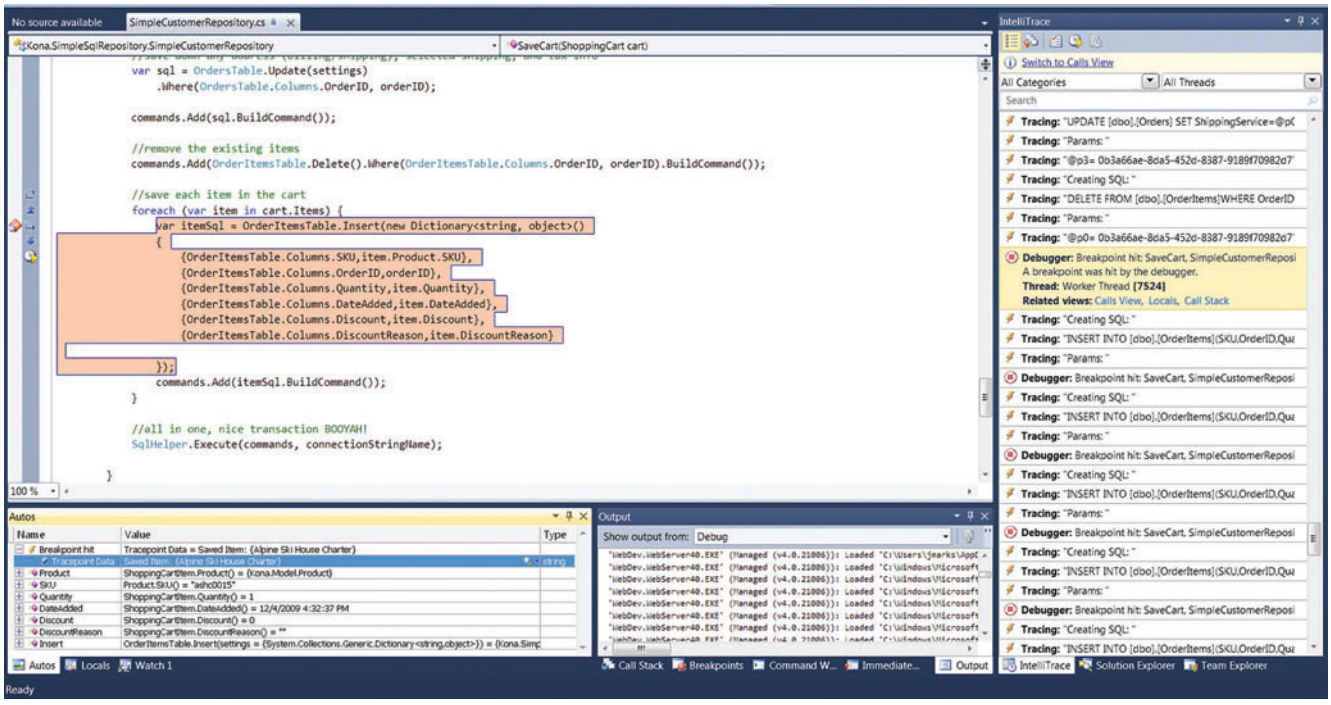


Figure 4 **Tracepoints Can Dynamically Add Trace Outputs to Code**

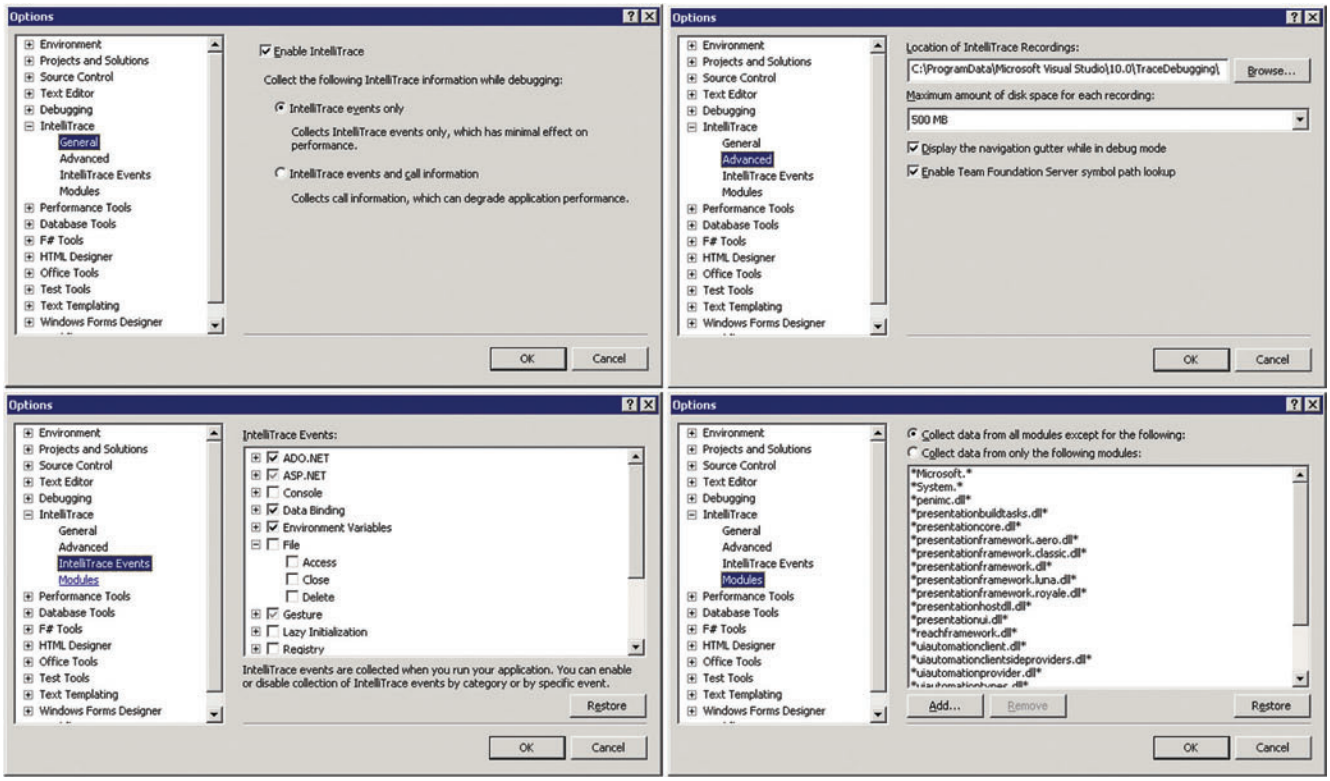


Figure 5 **IntelliTrace Settings Can Be Changed from the Options Dialog**

Turning up the Dial

By default, IntelliTrace is configured to collect only IntelliTrace events. This is a low-overhead solution but doesn't provide a full execution history of your application. When you need a deeper level of information, IntelliTrace can be configured to collect more data. Like other debugger settings, IntelliTrace can be configured from the Options dialog, accessible from the Tools menu (see **Figure 5**).

Choosing "IntelliTrace events and call information" configures IntelliTrace to collect not only the IntelliTrace events, but also call information at every method enter, exit and callsite, such as parameters and return values at those locations. Unfortunately, enabling this mode causes Edit-and-continue to be disabled during the debugging session. Obviously, your choice to collect more information means there's more overhead to your application. We worked very hard to find the balance between useful information and performance, but we do give you full control.

Every debugging session creates an IntelliTrace file stored on disk that is automatically cleaned up when Visual Studio closes. The IntelliTrace Advanced pane reached through the Options dialog lets you configure where you want the files created during debugging to be stored and how large they can be. If the maximum file size is reached, a circular buffer is used to help compress and truncate the information stored in the IntelliTrace log, thereby reducing the footprint on disk of the log file. The two other settings on the Advanced pane let you hide the navigation gutter and disable Team Foundation Server (TFS) lookup of available symbols.

Because of performance concerns, only a subset of the defined IntelliTrace events has been enabled for collection by default. Some of the events you may want to consider enabling are console output, file accesses, lazy initialization, registry accesses and threading events. You can enable or disable an entire category of events or individual events from the IntelliTrace Events pane in the Options dialog.

The final IntelliTrace Options pane allows you to control which modules IntelliTrace collects data from. All modules except those shipped by Microsoft as part of the Microsoft .NET Framework and Visual Studio are collected by default. A lot of data can be collected in this manner, so you may consider changing this list to an inclusion list and specifying only the modules you care about. Conversely, if you use some common third-party libraries, you may wish to exclude these, because they're out of your control.

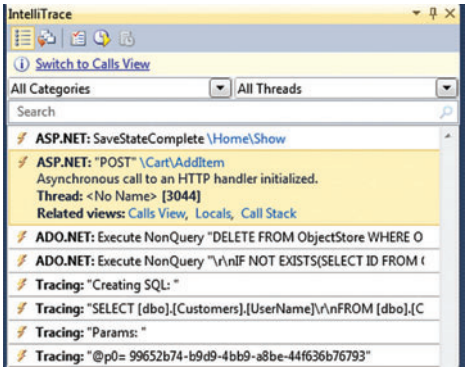


Figure 6 **The IntelliTrace Events View Can Help You Start Your Investigation**

Investigating a User Code Error

Now that I've verified that the "About us" page is operational in the demo application, let's make sure our shopping cart is working as well. When I add a paper airplane to the cart for purchase, I see that indeed it has been correctly added to the cart. However, when I repeat this action to

add a second instance of the item to the cart, the quantity still shows as 1, though I expected it to show 2. I want to use the debugger to solve the problem without having to restart, but the list of IntelliTrace events probably won't help me in this scenario (all the work was done in my code, not the .NET Framework). This is where the "IntelliTrace events and call information" mode of IntelliTrace can help show me an execution history of my application. Let's break into the debugger and see what additional features of IntelliTrace I can use to solve this problem.

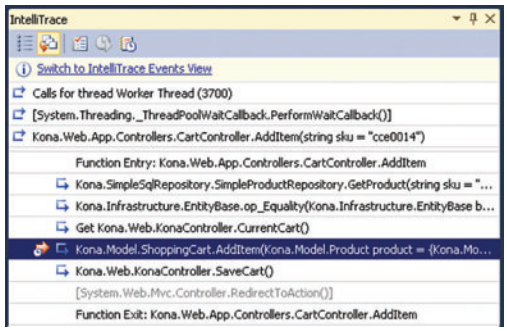


Figure 7 **The IntelliTrace Calls View Shows the Execution History of the Application**

I start with the hypothesis that something went wrong with my "Add Item to Cart" logic. Because this is a Model-View-Controller (MVC) application, there is no event handler for the button click, but rather a POST message has been sent and handled by the MVC. This POST message has been recorded as an IntelliTrace event and, although it isn't useful in itself, I use the event as a starting point for my investigation. By clicking this IntelliTrace event, I can jump to the point of the debugging session where the "Add Item" logic began. I can quickly find these POST messages by searching for the word "POST" in the IntelliTrace window. This action was taken twice by the user so, as I expected, two results are returned from the search. After selecting the second event, clearing the search field shows it in context with all of the collected events (see **Figure 6**).

Now that I have context on where I am in the application's timeline, I want to dig deeper and gain an understanding of the method calls that were made. From the Events view, I can switch views to see the execution history. By clicking on the "Switch to IntelliTrace Calls View" link at the top of the window, I can transition to the Calls view to see a complete execution history of the application (see **Figure 7**). At any time, I can go back to the Events view by clicking the "Switch to IntelliTrace Events View" link.

One mechanism for navigating the execution history is to use the Calls view to drill into the calls you're interested in investigating. Each time you double-click on a call in the bottom half of the Calls view, the call is popped to the bottom of the top half of the view, and the Instruction Pointer syncs in the code editor to the method entry point of the call, just like in live debugging when you walk up the call stack. You can continue to navigate and walk backward and forward through the data collected by IntelliTrace history in this way. Navigating through the Calls view is a quick mechanism for gaining an overview of the execution history and making large jumps around the code base.

Using the Calls view is only one method of navigation. I can also navigate by stepping through the code. In the gutter of the source code window, there's a new set of DVR-style controls that allow you to step through the code, just as in a traditional debugging session (see **Figure 8**). Because I'm in IntelliTrace debugging mode, stepping is by recorded events that occur at every callsite, function enter and function exit. Of course, F10/F11 work as expected if you prefer keyboard controls.

Figure 8 **The Navigation Bar Offers DVR-Style Controls to Let You Step Through Your Application**

These two navigation methods are great for investigation, but sometimes you know exactly where you would have set the breakpoint. In the case of this application, I know the name of the function where the item is added to the cart: Kona.Model.ShoppingCart::AddItem. What I really want to do is jump to the second time this function was called and inspect the values passed into and returned from the function. More specifically, I want to seek to the line of code where the "AdjustQuantity" function call was made. Of course, IntelliTrace also supports this navigation technique.

In the editor, I can right-click on the line I wish to seek to and choose "Search For This Line In IntelliTrace" from the context menu (see **Figure 9**). The search begins and the results are presented in a search bar at the top of the editor window, allowing me to navigate between the search results. After synchronizing to the second search result, my instruction pointer is right where I want it to be and I can investigate the call using the other debugger windows (see **Figure 10**).

Looking in the Locals window at this point, you can see that the cart is being adjusted so that the product's new quantity in the cart is 1. This is the bug; I was expecting the adjusted quantity to be 2. Looking at the line of code, the new Quantity parameter should

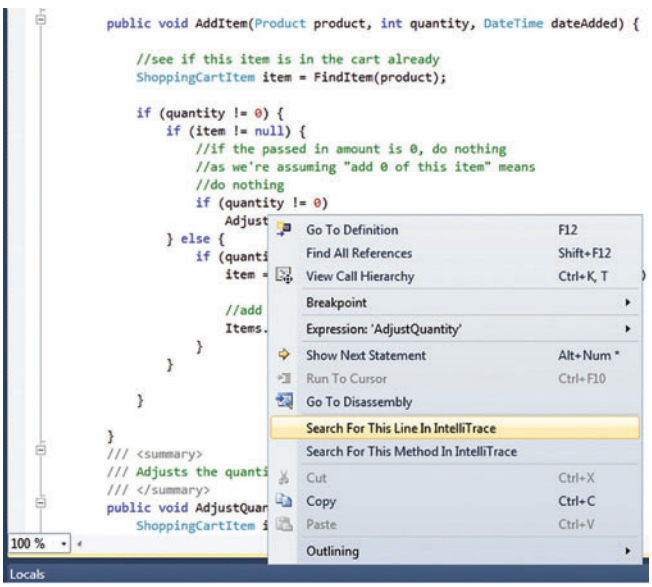


Figure 9 **The Search Functionality Lets You Jump Right to a Specific Function Call**

Now v4.0!

Why is Amyuni PDF so interesting?

Proven

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

High-Performance

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

Rapid Integration

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

Expertise

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

OEM Licenses

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

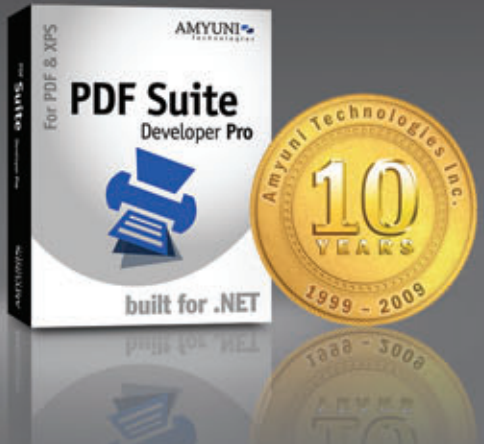
Customization

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

SD Times 2009 100

Windows 7

Certified for Windows Vista



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

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

www.amyuni.com

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

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

AMYUNI Technologies

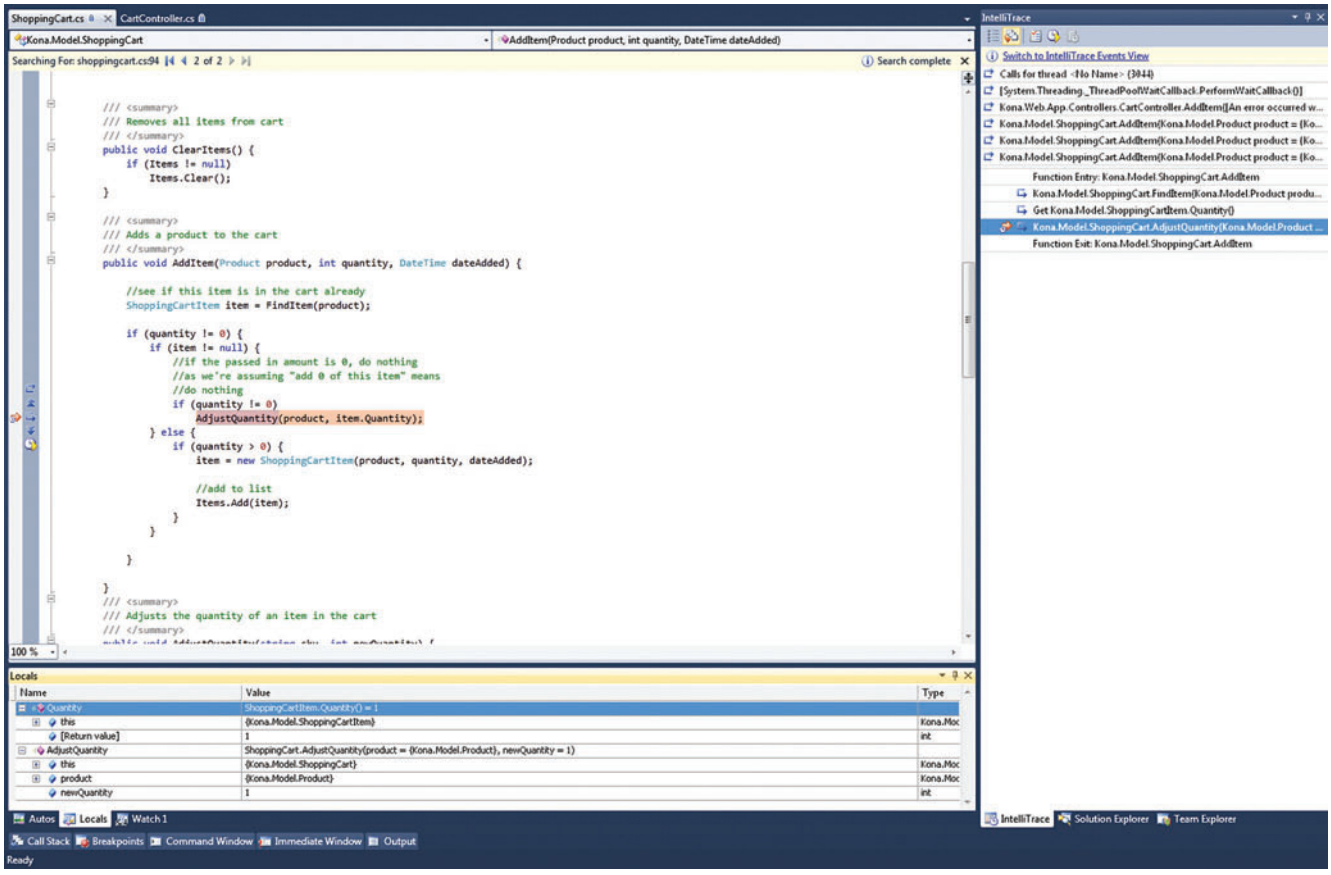


Figure 10 **Search Results Are Displayed in a Search Results Bar at the Top of the Editor Window**

take into account not only "item.Quantity" but the "quantity" variable passed into the function call as well. The fix is to change the function call to:

```
AdjustQuantity(product, item.Quantity + quantity);
```

Eliminating the Dreaded 'No Repro' Scenario

All too often, testers and developers participate in the "no repro" dance in which the tester files a bug stating something is broken, only to have it resolved back with a comment "it didn't repro on my machine." Neither the developer nor the tester wants to participate in the dance, but they don't have the right set of tools to help them communicate what happened at the point that the failure occurred. This is why we designed the system so that the same diagnostic information IntelliTrace provides to the developer during a debugging session can be collected during the execution of manual tests through the Microsoft Test Manager (MTM).

Let's revisit the first scenario I debugged, where the "About us" page failed to render correctly. If a tester filed this bug, it would probably have a title like "Application error when viewing the about page," and perhaps, if the developer was lucky, a screenshot of the

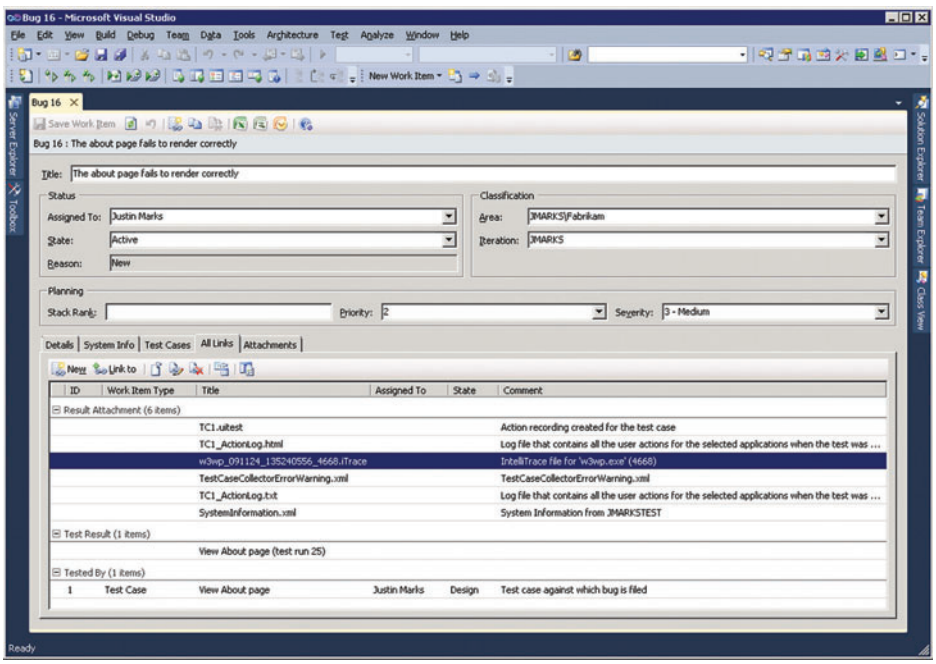


Figure 11 **Collected IntelliTrace Files Are Accessible Through the "All Links" Tab**

Tracking down memory leaks with ANTS Memory Profiler™

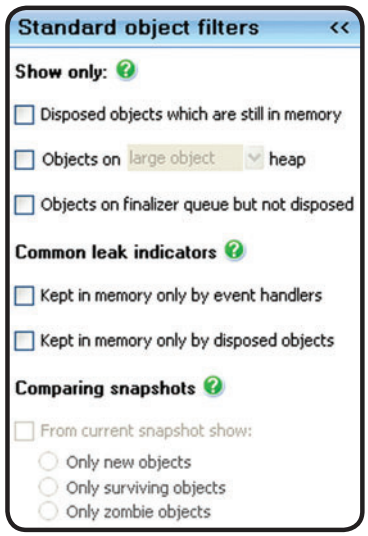
There are several possible strategies for finding memory leaks, but here is one of the most common ones:

1. Take a first snapshot of your application's memory usage to use as a baseline

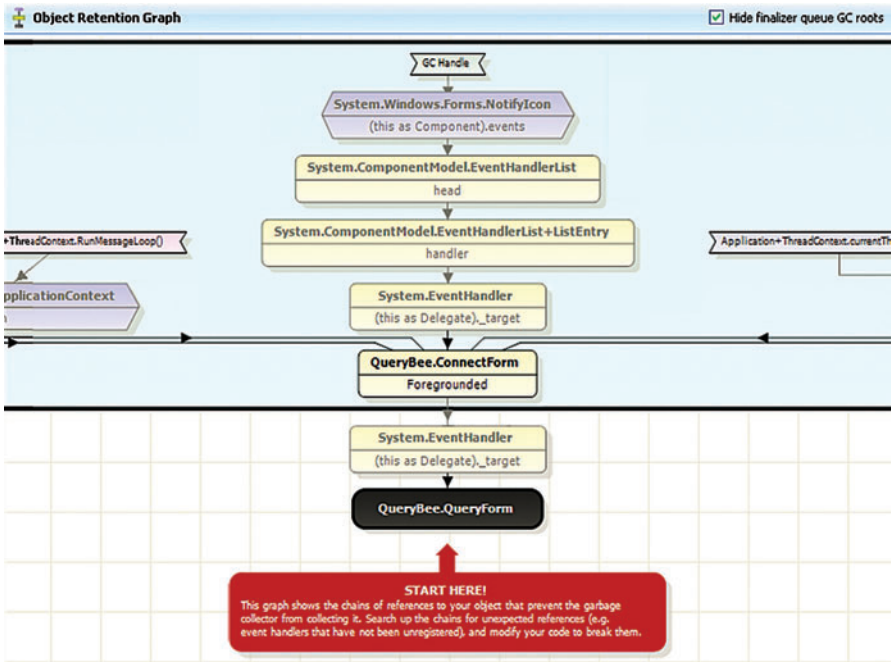
Once you start your profiling session, and before you delve deeper into your application, it's good practice to take a first snapshot of the heap memory, so you can have a baseline snapshot to compare others to.

2. Take a second snapshot and compare snapshot data

Now use your application as normal, performing the tasks which are causing the memory leak. You can then take your second snapshot and proceed to analyzing the differences between snapshots 1 and 2.



The Object Filters panel allows you to focus on those objects which are common causes of memory leaks.



The Object Retention Graph shows you what is still referencing your object.

3. Create an Object Retention Graph

Find out which objects have grown most in size, pick one, and create an Object Retention Graph which will show you what is still holding your object in memory. In the example above, QueryBee.QueryForm (the black object) has been closed, but is still referenced by an event handler. The Object Retention Graph is the key to finding where your memory leak lies.

4. Fix your code!

Once you figure out what is holding your object in memory, you need to go into your code and break the chain of references that is preventing your object from being garbage collected. For example, in this scenario above, we need to unregister the event handler.

Download your 14-day, fully functional free trial from www.red-gate.com.

Laila Lotfi *is a Brand Manager in the .NET Tools division at Red Gate Software. She works with the ANTS Profiler and .NET Reflector teams.*

error would be attached. If you received a bug like this today, how might you go about debugging it? Most likely, you would load the application's solution from source control and reproduce the problem on your developer machine. In this case, you would be able to debug the issue and solve the problem, but think of how much time this would take. Or what if the problem was caused by a difference in configuration between your machine and the tester's machine? Developers are much more productive when debugging issues rather than setting up repro environments.

With Visual Studio 2010, the MTM has given testers the ability to conveniently author, manage and execute both manual and automated tests. But that's not all the tool does. MTM offers testers the ability to have diagnostic data adapters collect information in the background while a test is executing. For example, you can automatically collect a video recording of the test being executed so the developer can see exactly what the tester saw. Other collectors include the ability to collect system information and Event Logs from the box being tested on.

We've added an IntelliTrace diagnostic data adapter to automatically collect IntelliTrace files in the background. When the tester fails a test step and chooses to file a bug, the collected IntelliTrace file is automatically uploaded to TFS and linked to the bug. This gives the developer a much richer set of information to debug with than simply a set of repro steps. When the developer opens the IntelliTrace file linked to the bug, he's given an experience similar to debugging a mini-dump, but over the entire timeline of the application, not just a point in time when the application crashed. IntelliTrace data can be collected for any managed application on local or remote test agents, including ASP.NET applications running under IIS.

When I open a TFS bug, the MTM has automatically added repro steps based on the manual test steps described by the tester. What's even cooler is that if I look at the "All Links" tab of the bug (see **Figure 11**), I can see that an IntelliTrace file was also collected. By double-clicking on this file, I'm brought to a summary page view of the IntelliTrace file (see **Figure 12**).

The summary page gives me a lot of information about what the IntelliTrace file contains. At the top of the page, I'm given a timeline view of the threads in the application. If I expand the "Threads List" section of the summary page, I can see the same data in a tabular format. Double-clicking on a thread will start a debugging session from the IntelliTrace file and set my instruction pointer to the beginning of the thread.

I'm also given a list of all the exceptions collected in the IntelliTrace file. If I select an exception, a vertical orange line appears on the thread timeline showing where the exception occurs in time. Double-clicking an exception starts an IntelliTrace debugging session and the exception event will be selected in the Intelli-

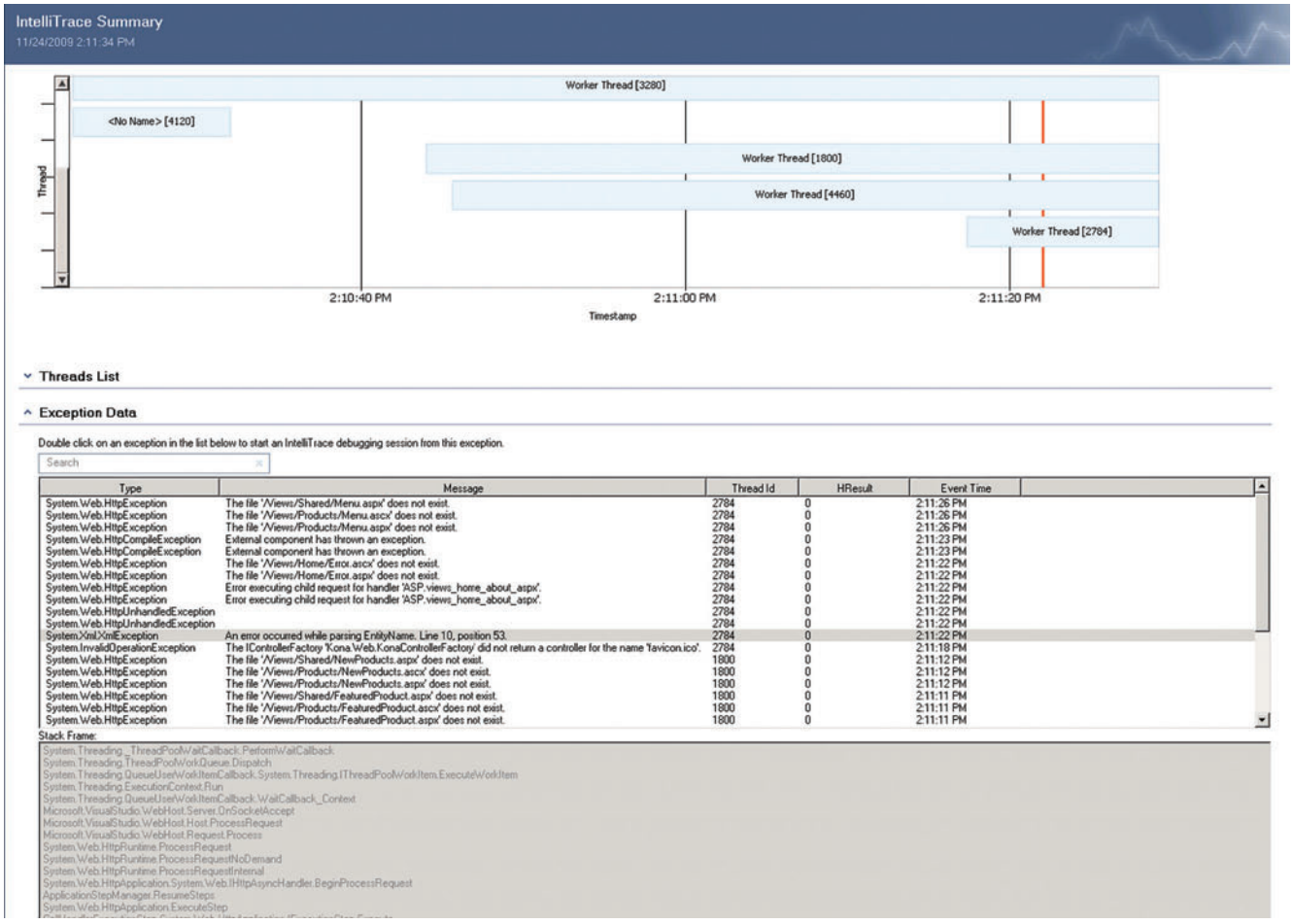


Figure 12 **The IntelliTrace Summary Provides an Overview of What Data Was Collected**

MICRO FOCUS®

Leading the Evolution™

Next Generation Developer Tools

@VS2010 LAUNCH

APRIL 12TH–14TH

BELLAGIO – LAS VEGAS

www.microfocus.com/VS2010

DEVELOP, ANALYZE AND TEST THE HIGHEST QUALITY CODE

Find Micro Focus at Stand 113

Microsoft® Visual Studio®

PARTNER

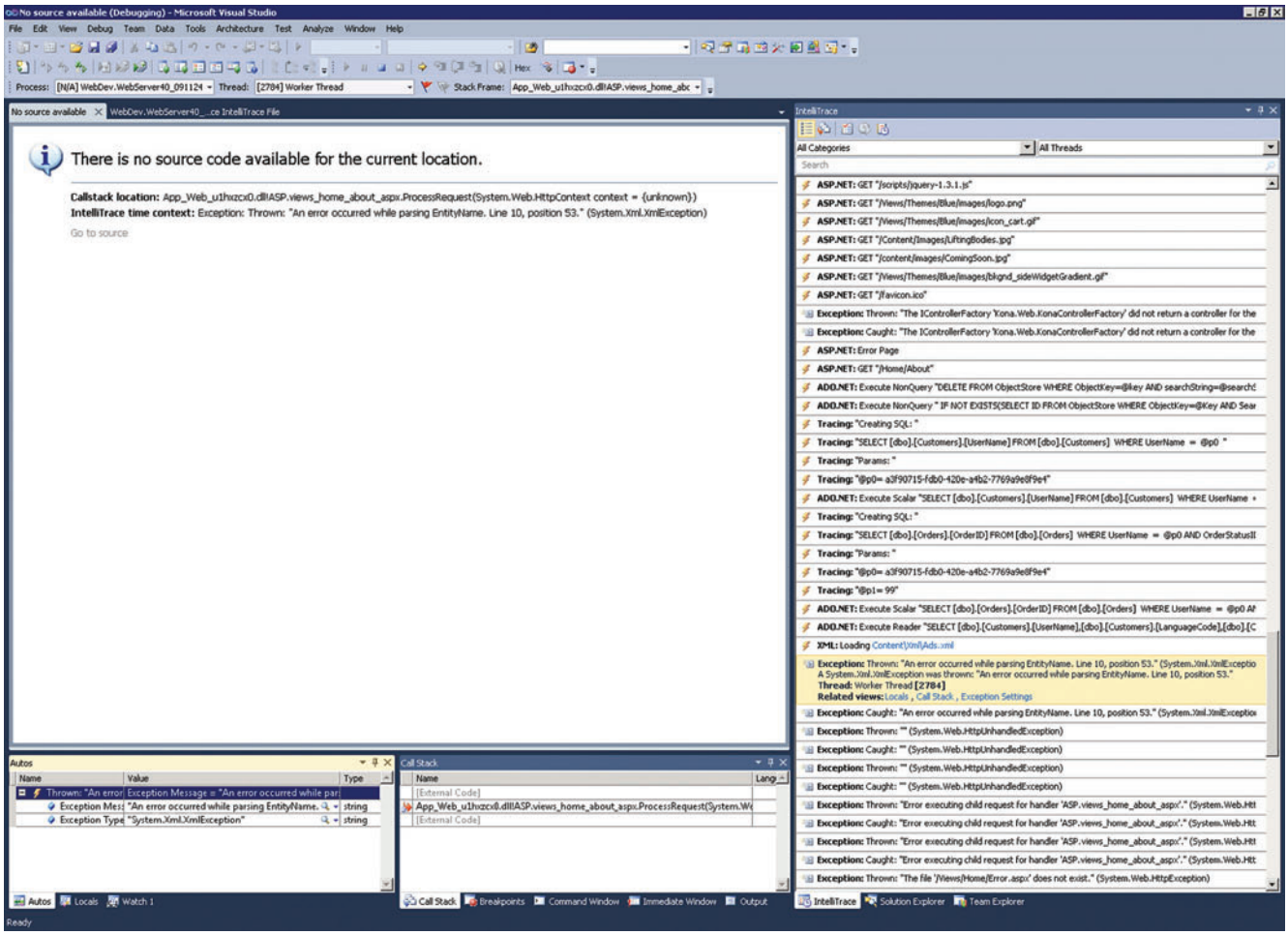


Figure 13 **When Debugging Starts from the Summary Page, Visual Studio Operates Just Like a Live Debugging Session**

Trace Events view. Further down on the summary page, I can see a list of all the test events matching up to the manual repro steps. The outcome of each step is also displayed. Clicking on one of these events also puts a vertical line on the thread timeline, and double-clicking an event starts an IntelliTrace debugging session selecting the closest event possible. The bottom of the summary page shows a list of system information collected about the computer upon which the tested application was running, and below that is a list of all the modules that were loaded into the process along with their file path.

You may be asking yourself, does this work on release builds? Of course! In addition, you don't need access to symbols when collecting or viewing the IntelliTrace data. Symbols are necessary only to link the collected data to source files, so they're helpful when you open the debug session.

Visual Studio 2010 and TFS have added support for symbol and source server, so if your application was built as part of the TFS build system, the right version of symbols and source code will automatically be downloaded from TFS as needed by IntelliTrace. You don't have to even know the symbol server path.

Once you start a debugging session from the summary page, the debugger works just as it would for a live debugging session (see **Figure 13**). Most debugger windows are available to you, and you can navigate to any of the data IntelliTrace has collected.

Wrapping Up

In this article, you've seen how IntelliTrace can greatly improve both your day-to-day development activities and your ability to quickly and easily diagnose problems without having to restart your application and debug with the traditional break-step-inspect technique. I've also shown you how an organization can reduce the number of "no repro" bugs by collecting IntelliTrace data during testing, enabling developers to debug issues offline without access to a live repro. This is just a brief introduction to the feature, and as you become more familiar with the power of IntelliTrace, it will start to change the way you debug. ■

JUSTIN MARKS *started at Microsoft in 2002 after receiving his BS in Computer Science and Engineering from MIT. He has worked on MSN.com as a systems engineer, Windows as a software design engineer in Test, and now Visual Studio as a program manager. As a PM on the Diagnostics team, Marks has been working on the IntelliTrace feature for the next release of Visual Studio 2010.*

THANKS to the following technical experts for reviewing this article:
Bill Boris, David Gray, Habib Heydarian and John Robbins

GET THE FASTEST CONTROLS... THAT LOOK GOOD TOO

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

Infragistics
Killer Apps. No Excuses.

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


Alien approved for 20+ Light Years!
THE CONTROLS HAVE LANDED.
113
MPH
SPORT
TRIP A
5001.2 M
DATA
Drive C
Program Files
Visual Studio 2008
ComponentOne Studio
Bin
PhotoShop
Corel 11
Drive D
My Images
My Music
NET
LOCAL
2010
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
USERCODE
BAND
LASTNAME
FIRSTNAME
CONTACTED
COMPANY
CUSTTYPE
NoFilter
Contains
NotContain
BeginsWith
EndsWith
Equals
NotEqual
IsEmpty
NotIsEmpty
Greater
GreaterOrEqual
Less
LessOrEqual
100's OF CONTROLS, 7 PLATFORMS
Featuring WinForms • WPF • ASP.NET • Silverlight • iPhone • Mobile • ActiveX
Grids • Charting • Reporting • Scheduling • Menus and Toolbars • Ribbon • Data Input • Editors • PDF

An Introduction to Functional Programming for .NET Developers

Chris Marinos

By now, there's a good chance you've heard about F#, the latest addition to the Microsoft Visual Studio language family. There are many exciting reasons to learn F#—its clean syntax, its powerful multi-threading capabilities and its fluid interoperability with other Microsoft .NET Framework languages. However, F# includes some important new concepts you'll need to understand before you can take advantage of these features.

A whirlwind tour is a good way to start learning another object-oriented language, or even a dynamic language like Ruby or Python. That's because you already know most of the vocabulary and you just need to learn new syntax. F# is different, though. F# is a functional programming language, and with that comes more new vocabulary than you may expect. Moreover, functional languages have traditionally been used in academia, so the definitions for these new terms can be difficult to understand.

Fortunately, F# is not designed to be an academic language. Its syntax allows you to use functional techniques to solve problems in new and better ways while still supporting the object-oriented and imperative styles that you're accustomed to as a .NET developer. Unlike other .NET languages, F#'s multi-paradigm structure means that you are free to choose the best style of programming for the problem that you're trying to solve. Functional programming in F# is about writing concise, powerful code to solve practical software problems. It's about using techniques like higher order functions and function composition to create powerful and easy to understand behaviors. It's also about making your code easier to understand, test and parallelize by removing hidden complexities.

But in order for you to take advantage of all of these fantastic features of F#, you need to understand the basics. In this article, I'll explain these concepts using vocabulary that you are already familiar with as a .NET developer. I will also show you some functional programming techniques that you can apply to your existing code and some ways in which you are already programming functionally. By the end of the article, you'll know enough about functional programming so that you can hit the ground running with F# in Visual Studio 2010.

This article discusses a prerelease version of Visual Studio 2010. All information is subject to change.

This article discusses:

- Functional and imperative programming
- Type inference and side effects
- Composing functions
- Currying and partial application

Technologies discussed:

F#, Visual Studio 2010

Code download available at:

code.msdn.microsoft.com/mag201004FSharp

Figure 1 **Side Effects of Mutable Variables**

```
public MemoryStream GetStream() {
  var stream = new MemoryStream();
  var writer = new StreamWriter(stream);
  writer.WriteLine("line one");
  writer.WriteLine("line two");
  writer.WriteLine("line three");
  writer.Flush();
  stream.Position = 0;
  return stream;
}

[TestMethod]
public void CausingASideEffect() {
  using (var reader = new StreamReader(GetStream())) {
    var line1 = reader.ReadLine();
    var line2 = reader.ReadLine();

    Assert.AreNotEqual(line1, line2);
  }
}
```

Functional Programming Basics

For most .NET developers, it's easier to understand what functional programming is by understanding what it isn't. Imperative programming is a style of programming that is considered to be the opposite of functional programming. It's also the style of programming you are probably most familiar with because most mainstream programming languages are imperative.

Functional programming and imperative programming differ on a very fundamental level, and you can see this in even the simplest code:

```
int number = 0;
number++;
```

This obviously increments a variable by one. That's not very exciting, but consider a different way that you could solve the problem:

```
const int number = 0;
const int result = number + 1;
```

The number is still incremented by one, but it's not modified in place. Instead, the result is stored as another constant because the compiler does not allow you to modify the value of a constant. You would say that constants are immutable because you can't change their values once they are defined. Conversely, the number variable from my first example was mutable because you can modify its value. These two approaches show one of the fundamental differences between imperative programming and functional programming. Imperative programming emphasizes the use of mutable variables whereas functional programming uses immutable values.

Most .NET developers would say that number and result in the previous example are variables, but as a functional programmer you need to be more careful. After all, the idea of a constant variable is confusing at best. Instead, functional programmers say that number and result are values. Make sure you reserve the term variable for objects that are mutable. Note that these terms are not exclusive to functional programming, but they are a lot more important when programming in a functional style.

This distinction may seem small, but it's the foundation for a lot of the concepts that make functional programming so powerful. Mutable variables are the root cause of a lot of nasty bugs. As you will see below, they lead to implicit dependencies between different parts of your code, which makes for many problems, especially related to concurrency. In contrast, immutable variables introduce significantly less complexity. They lead to functional techniques like using functions as values and compositional programming which I'll also explore in more detail later.

If you're skeptical of functional programming at this point, don't worry. That's natural. Most imperative programmers are trained to believe that you can't do anything useful with immutable values. However, consider this example:

```
string stringValue = "world!";
string result = stringValue.Insert(0, "hello ");
```

The Insert function built the "hello world!" string, but you know that Insert doesn't modify the source string's value. That's because strings are immutable in .NET. The designers of the .NET Framework used a functional approach because it made it easier to write better code with strings. Because strings are among the most widely used data types in the .NET Framework (along with other base types, like integers, DateTimes and so on), there's a good chance you've done more useful functional programming than you realize.

Putting F# to Work

F# comes with Visual Studio 2010, and you can find the latest version at msdn.microsoft.com/vstudio. If you use Visual Studio 2008, you can download an F# add-in from the F# Developer Center at msdn.microsoft.com/fsharp, where you'll also find installation instructions for Mono.

F# adds a new window to Visual Studio called F# Interactive that, unsurprisingly, allows you to interactively execute F# code. You can think of it as a more powerful version of the Immediate Window that you can access even when you're not in debug mode. If you're familiar with Ruby or Python, you'll recognize that F# Interactive is a Read-Evaluate-Print Loop (REPL), which is a helpful tool to have for learning F# and quickly experimenting with code.

> F# adds a new window to Visual Studio called F# Interactive.

I'll use F# Interactive in this article to show you what happens when example code is compiled and run. If you highlight code in Visual Studio and press Alt+Enter, you send the code to F# Interactive. To see this, here is the simple addition example in F#:

```
let number = 0
let result = number + 1
```

When you run this code in F# Interactive, you get the following:

```
val number : int = 0
val result : int = 1
```

You can probably guess by the term val that number and result are both immutable values, not mutable variables. You can see this by using <-, the F# assignment operator:

```
> number <- 15;;

  number <- 15;;
  ^^^^^^^^^^^

stdin(3,1): error FS0027: This value is not mutable
>
```

Because you know that functional programming is based on immutability, this error should make sense. The let keyword is used to create immutable bindings between names and values. In C# terms,

Figure 2 **Mutable PiggyBanks**

```
public class PiggyBank{
  public PiggyBank(int coins){
    Coins = coins;
  }

  public int Coins { get; set; }
}

private void DepositCoins(PiggyBank piggyBank){
  piggyBank.Coins += 10;
}

private void BuyCandy(PiggyBank piggyBank){
  if (piggyBank.Coins < 7)
    throw new ArgumentException(
      "Not enough money for candy!", "piggyBank");

  piggyBank.Coins -= 7;
}
```

everything is const by default in F#. You can make a mutable variable if you want, but you have to explicitly say so. The defaults are just the opposite of what you're familiar with in imperative languages:

```
let mutable myVariable = 0
myVariable <- 15
```

Type Inference and Whitespace Sensitivity

F# lets you declare variables and values without specifying their type, so you might assume that F# is a dynamic language, but that's not true. It is important to understand that F# is a static language just like C# or C++. However, F# has a powerful type inference system that allows you to avoid specifying the types of objects in many places. This allows for a simple and succinct syntax, while still providing the type safety of static languages.

Although type inference systems like this aren't really found in imperative languages, type inference isn't directly related to functional programming. However, type inference is a critical notion to understand if you want to learn F#. Fortunately, if you're a C# developer, chances are you're already familiar with basic type inference because of the var keyword:

```
// Here, the type is explictily given
Dictionary<string, string> dictionary =
  new Dictionary<string, string>();

// but here, the type is inferred
var dictionary = new Dictionary<string, string>();
```

Both lines of C# code create new variables that are statically typed as Dictionary<string, string>, but the var keyword tells the complier to infer the type of the variable for you. F# takes this concept to the next level. For example, here is an add function in F#:

```
let add x y =
    x + y

let four = add 2 2
```

There isn't a single type annotation in the above code, but F# Interactive reveals the static typing:

```
val add : int -> int -> int
val four : int = 4
```

I'll explain the arrows in more detail later, but for now you can interpret this to mean that add is defined to take two int arguments, and that four is an int value. The F# compiler was able to infer this based on the way add and four were defined. The rules the complier uses to do this are beyond the scope of this article, but you can learn more about them at the F# Developer Center if you're interested.

Type inference is one way that F# reduces noise in your code, but notice that there are no curly braces or keywords to denote the body or return value of the *add* function. That's because F# is a whitespace-sensitive language by default. In F#, you indicate the body of a function by indentation, and you return a value by making sure that it is the last line in the function. Like type inference, whitespace sensitivity has no direct relationship to functional programming, but you need to be familiar with the concept in order to use F#.

Side Effects

Now you know that functional programming is different from imperative programming because it relies on immutable values instead of mutable variables, but that fact isn't very useful by itself. The next step is to understand side effects.

In imperative programming, a function's output depends on its input argument and the current state of the program. In functional programming, functions only depend on their input arguments. In other words, when you call a function more than once with the same input value, you always get the same output value. The reason this isn't true in imperative programming is because of side effects, as demonstrated in **Figure 1**.

On the first call to ReadLine, the stream gets read until a new line is encountered. Then, ReadLine returns all of the text up to the new line. In between those steps, a mutable variable representing the stream's position gets updated. That's the side effect. On the second call to ReadLine, the value of the mutable position variable has changed, so ReadLine returns a different value.

Now let's look at one of the most significant consequences of using side effects. First, consider a simple PiggyBank class and some methods to work with it (see **Figure 2**).

> In F#, you focus on evaluating functions for their result values instead of their side effects.

If you have a piggy bank with 5 coins in it, you can call DepositCoins before BuyCandy, but reversing the order raises an exception:

```
// this works fine
var piggyBank = new PiggyBank(5);

DepositCoins(piggyBank);
BuyCandy(piggyBank);

// but this raises an ArgumentException
var piggyBank = new PiggyBank(5);

BuyCandy(piggyBank);
DepositCoins(piggyBank);
```

The BuyCandy function and the DepositCoins function both update the state of the piggy bank through the use of a side effect. Consequently, the behavior of each function depends on the state of the piggy bank. Because the number of coins is mutable, the order in which these functions execute is significant. In other words, there is an implicit timing dependency between these two methods.

Now let's make the number of coins read only to simulate an immutable data structure. **Figure 3** shows that BuyCandy and DepositCoins now return new PiggyBank objects instead of updating an existing PiggyBank.

As before, if you try to call BuyCandy before DepositCoins, you will get an argument exception:

```
// still raises an ArgumentException
var piggyBank = new PiggyBank(5);

BuyCandy(piggyBank);
DepositCoins(piggyBank);
```

But now, even if you revert the order, you'll get the same result:

```
// now this raises an ArgumentException,  too!
var piggyBank = new PiggyBank(5);

DepositCoins(piggyBank);
BuyCandy(piggyBank);
```

Here, BuyCandy and DepositCoins only depend on their input argument because the number of coins is immutable. You can execute the functions in any order and the result is the same. The implicit time dependency is gone. However, since you probably want BuyCandy to succeed, you need to make the result of BuyCandy depend on the output of DepositCoins. You need to make the dependency explicit:

```
var piggyBank = new PiggyBank(5);
BuyCandy(DepositCoins(piggyBank));
```

This is a subtle difference with far-reaching consequences. Shared mutable state and implicit dependencies are the source of some of the most diabolical bugs in imperative code, and they're the reason that multi-threading is so difficult in imperative languages. When you have to worry about the order in which functions execute, you need to rely on cumbersome locking mechanisms to keep things straight. Pure functional programs are free of side effects and implicit time dependencies, so the order in which functions execute doesn't matter. This means you don't have to worry about locking mechanisms and other error-prone multi-threading techniques.

Easier multi-threading is a major reason that functional programming is getting attention lately, but there are many other benefits of programming in a functional style. Side effect-free functions are easier to test because each function relies only on its input arguments. They are easier to maintain because they don't implicitly rely on logic from other setup functions. Side effect-free functions also tend to be smaller and easier to combine. I'll cover this last point in more detail shortly.

In F#, you focus on evaluating functions for their result values instead of their side effects. In imperative languages, it is common to call a function to do something; in functional languages, functions are called to yield a result. You can see this in F# by looking at the if statement:

```
let isEven x =
    if x % 2 = 0 then
        "yes"
    else
        "no"
```

You know that in F#, the last line of a function is its return value, but in this example, the last line of the function is the if statement.

Figure 3 **Immutable PiggyBanks**

```
public class PiggyBank{
  public PiggyBank(int coins){
    Coins = coins;
  }

  public int Coins { get; private set; }
}

private PiggyBank DepositCoins(PiggyBank piggyBank){
  return new PiggyBank(piggyBank.Coins + 10);
}

private PiggyBank BuyCandy(PiggyBank piggyBank){
  if (piggyBank.Coins < 7)
    throw new ArgumentException(
      "Not enough money for candy!", "piggyBank");

  return new PiggyBank(piggyBank.Coins - 7);
}
```

This isn't a compiler trick. In F#, even if statements are designed to return values:

```
let isEven2 x =
    let result =
        if x % 2 = 0 then
            "yes"
        else
            "no"
    result
```

The result value is of type string, and it is assigned directly to the if statement. It's similar to the way the conditional operator works in C#:

```
string result = x % 2 == 0 ? "yes" : "no";
```

The conditional operator emphasizes returning a value over causing a side effect. It's a more functional approach. In contrast, the C# if statement is more imperative because it does not return a result. All it can do is cause side effects.

Composing Functions

Now that you've seen some of the benefits of side-effect-free functions, you're ready to use functions to their full potential in F#. First, let's start with some C# code to take the square of the numbers zero through 10:

```
IList<int> values = 0.Through(10).ToList();

IList<int> squaredValues = new List<int>();

for (int i = 0; i < values.Count; i++) {
  squaredValues.Add(Square(values[i]));
}
```

Aside from the Through and Square helper methods, this code is fairly standard C#. Good C# developers would probably take umbrage with my use of a for loop instead of a foreach loop, and rightly so. Modern languages like C# offer foreach loops as an abstraction to make walking through enumerations easier by removing the need for explicit indexers. They succeed in this goal, but consider the code in **Figure 4**.

The foreach loops in this example are similar, but each loop body performs a slightly different operation. Imperative programmers have traditionally been okay with this code duplication because it's considered to be idiomatic code.

Functional programmers take a different approach. Instead of creating abstractions like foreach loops to help walk lists, they use side effect-free functions:

```
let numbers = {0..10}
let squaredValues = Seq.map Square numbers
```

This F# code also squares a sequence of numbers, but it does so using a higher-order function. Higher-order functions are simply functions that accept another function as an input argument. In this case, the function Seq.map accepts the Square function as an argument. It applies this function to each number in the numbers sequence and returns the sequence of squared numbers. Higher-order functions are why many people say functional programming uses functions as data. This just means that functions can be used as parameters or assigned to a value or variable just like an int or a string. In C# terms, it's very similar to the concepts of delegates and lambda expressions.

Higher order functions are one of the techniques that makes functional programming so powerful. You can use higher-order functions to isolate the duplicated code inside of foreach loops and encapsulate it into standalone, side effect-free functions. These functions each perform one small operation that the code inside a foreach loop would have handled. Because they are side effect-free, you can combine these functions to create more readable, easier-to-maintain code that accomplishes the same thing as foreach loops:

```
let squareOfEvens =
    numbers
    |> Seq.filter IsEven
    |> Seq.map Square
```

The only confusing part about this code may be the |> operator. This operator is used to make code more readable by allowing you to reorder the arguments to a function so that the last argument is the first thing that you read. Its definition is very simple:

```
let (|>) x f = f x
```

Without the |> operator, the squareOfEvens code would look like this:

```
let squareOfEvens2 =
  Seq.map Square (Seq.filter IsEven numbers)
```

If you use LINQ, employing higher-order functions in this way should seem very familiar. That's because LINQ is deeply rooted in functional programming. In fact, you can easily translate the square of evens problem into C# using methods from LINQ:

```
var squareOfEvens =
  numbers
  .Where(IsEven)
  .Select(Square);
```

This translates to the following LINQ query syntax:

```
var squareOfEvens = from number in numbers
  where IsEven(number)
  select Square(number);
```

Using LINQ in C# or Visual Basic code allows you to exploit some of the power of functional programming on an everyday basis. It's a great way to learn functional programming techniques.

When you start to use higher-order functions on a regular basis, you will eventually come across a situation in which you want to create a small, very specific function to pass into a higher-order function. Functional programmers use lambda functions to solve this problem. Lambda functions are simply functions you define without giving them a name. They are normally small and have a very specific use. For example, here is another way that you could square even numbers using a lambda:

```
let withLambdas =
    numbers
    |> Seq.filter (fun x -> x % 2 = 0)
    |> Seq.map (fun x -> x * x)
```

The only difference between this and the previous code to square even numbers is that the Square and IsEven are defined as lambdas. In F#, you declare a lambda function using the fun keyword. You should only use lambdas to declare single-use functions because they can't easily be used outside the scope in which they are defined. For this reason, Square and IsEven are poor choices for lambda functions because they are useful in many situations.

Currying and Partial Application

You now know almost all of the basics you need to start working with F#, but there is one more concept you should be familiar with. In previous examples, the |> operator and the arrows in type signatures from F# Interactive are both tied to a concept known as currying.

Currying means breaking a function with many arguments into a series of functions that each take one argument and ultimately produce the same result as the original function. Currying is probably the most challenging topic in this article for a .NET developer, particularly because it is often confused with partial application. You can see both at work in this example:

```
let multiply x y =
    x * y

let double = multiply 2
let ten = double 5
```

Right away, you should see behavior that is different from most imperative languages. The second statement creates a new function called double by passing one argument to a function that takes two. The result is a function that accepts one int argument and yields the same output as if you had called multiply with x equal to 2 and y equal to that argument. In terms of behavior, it's the same as this code:

```
let double2 z = multiply 2 z
```

Often, people mistakenly say that multiply is curried to form double. But this is only somewhat true. The multiply function is curried, but that happens when it is defined because functions in F# are curried by default. When the double function is created, it's more accurate to say that the multiply function is partially applied.

Figure 4 **Using foreach Loops**

```
IList<int> values = 0.Through(10).ToList();

// square a list
IList<int> squaredValues = new List<int>();

foreach (int value in values) {
  squaredValues.Add(Square(value));
}

// filter out the even values in a list
IList<int> evens = new List<int>();

foreach(int value in values) {
  if (IsEven(value)) {
    evens.Add(value);
  }
}

// take the square of the even values
IList<int> results = new List<int>();

foreach (int value in values) {
  if (IsEven(value)) {
    results.Add(Square(value));
  }
}
```

Let's go over these steps in more detail. Currying breaks a function with many arguments into a series of functions that each take one argument and ultimately produce the same result as the original function. The multiply function has the following type signature according to F# Interactive:

```
val multiply : int -> int -> int
```

Up to this point, you decrypted this to mean that multiply is a function that takes two int arguments and returns an int result. Now I'll explain what really happens. The multiply function is really a series of two functions. The first function takes one int argument and returns another function, effectively binding x to a specific value. This function also accepts an int argument that you can think of as the value to bind to y. After calling this second function, x and y are both bound, so the result is the product of x and y as defined in the body of double.

> Currying means breaking a function with many arguments into a series of functions.

To create double, the first function in the chain of multiply functions is evaluated to partially apply multiply. The resulting function is given the name double. When double is evaluated, it uses its argument along with the partially applied value to create the result.

Using F# and Functional Programming

Now that you have enough vocabulary to get started with F# and functional programming, you have plenty of options for what to do next.

F# Interactive allows you to explore F# code and quickly build up F# scripts. It is also useful for validating everyday questions about the behavior of .NET library functions without resorting to help files or Web searches.

F# excels at expressing complicated algorithms, so you can encapsulate these portions of your applications into F# libraries that can be called from other .NET languages. This is especially useful in engineering applications or multi-threaded situations.

Finally, you can use functional programming techniques in your everyday .NET development without even writing F# code. Use LINQ instead of for or foreach loops. Try using delegates to create higher-order functions. Limit your use of mutability and side effects in your imperative programming. Once you start writing code in a functional style, you'll soon find yourself wishing you were writing more F# code. ■

CHRIS MARINOS *is a software consultant at SRT Solutions in Ann Arbor, Mich. You can hear him talk about F#, functional programming, and other exciting topics at events around the Ann Arbor area or on his blog at srtsolutions.com/blogs/chrismarinos.*

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

Exploring New C++ and MFC Features in Visual Studio 2010

Sumit Kumar

Visual Studio 2010 presents huge benefits for C++ developers. From the ability to employ the new features offered by Windows 7 to the enhanced productivity features for working with large code bases, there is something new and improved for just about every C++ developer.

In this article, I will explain how Microsoft has addressed some of the broad problems faced by C++ developers. Specifically, Visual Studio 2010 enables a more modern programming model by adding core language features from the upcoming C++0x standard, and by overhauling the standard library to take advantage of the new language features. There are new parallel programming libraries and tools to simplify the creation of parallel programs. You'll also find enhanced overall performance and developer productivity thanks to IntelliSense and code-understanding features that scale to large code bases. And you'll benefit from the improved performance of libraries and other features across design time, build time, compile time and link time.

Visual Studio 2010 migrates the build system to MSBuild to make it more customizable and to support native multi-targeting. And enhancements in the MFC library harness the power of new Windows 7 APIs, enabling you to write great Windows 7 applications.

Let's take a closer look at these C++-focused advancements in Visual Studio 2010.

This article discusses a prerelease version of Visual Studio 2010. All information is subject to change.

This article discusses:

- C++0x core language features
- C++ library improvements
- Concurrent programming
- MSBuild integration
- Building apps for Windows 7

Technologies discussed:

Visual Studio 2010, Visual C++, MFC

C++0x Core Language Features

The next C++ standard is inching closer to being finalized. To help you get started with the C++0x extensions, the Visual C++ compiler in Visual Studio 2010 enables six C++0x core language features: lambda expressions, the auto keyword, rvalue references, static_assert, nullptr and decltype.

Lambda expressions implicitly define and construct unnamed function objects. Lambdas provide a lightweight natural syntax to define function objects where they are used, without incurring performance overhead.

Function objects are a very powerful way to customize the behavior of Standard Template Library (STL) algorithms, and can encapsulate both code and data (unlike plain functions). But function objects are inconvenient to define because of the need to write entire classes. Moreover, they are not defined in the

place in your source code where you're trying to use them, and the non-locality makes them more difficult to use. Libraries have attempted to mitigate some of the problems of verbosity and non-locality, but don't offer much help because the syntax becomes complicated and the compiler errors are not very friendly. Using function objects from libraries is also less efficient since the function objects defined as data members are not in-lined.

Lambda expressions address these problems. The following code snippet shows a lambda expression used in a program to remove integers between variables x and y from a vector of integers.

```
v.erase(remove_if(v.begin(),
   v.end(), [x, y](int n) {
   return x < n && n < y; }),
   v.end());
```

The second line shows the lambda expression. Square brackets, called the lambda-introducer, indicate the definition of a lambda expression. This lambda takes integer n as a parameter and the lambda-generated function object has the data members x and y. Compare that to an equivalent handwritten function object to get an appreciation of the convenience and time-saving lambdas provide:

```
class LambdaFunctor {
public:
  LambdaFunctor(int a, int b) : m_a(a), m_b(b) { }
  bool operator()(int n) const {
    return m_a < n && n < m_b; }
private:
  int m_a;
  int m_b;
};
v.erase(remove_if(v.begin(), v.end(),
  LambdaFunctor(x, y)), v.end());
```

The auto keyword has always existed in C++, but it was rarely used because it provided no additional value. C++0x repurposes this keyword to automatically determine the type of a variable from its initializer. Auto reduces verbosity and helps important code to stand out. It avoids type mismatches and truncation errors. It also helps make code more generic by allowing templates to be written that care less about the types of intermediate expressions and effectively deals with undocumented types like lambdas. This code shows how auto saves you from typing the template type in the for loop iterating over a vector:

```
vector<int> v;
for (auto i = v.begin(); i != v.end(); ++i) {
// code
}
```

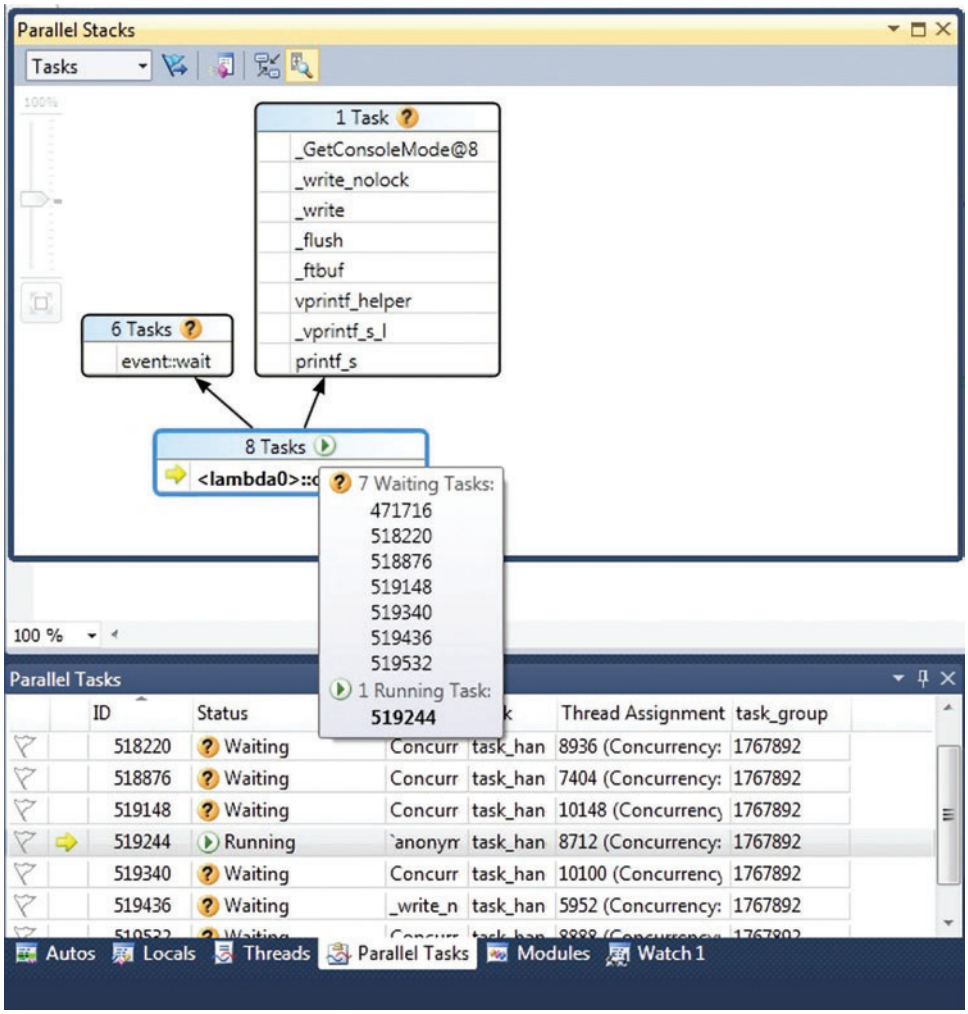


Figure 1 **Parallel Stacks and Parallel Tasks Debug Windows**

Rvalue references are a new reference type introduced in C++0x that help solve the problem of unnecessary copying and enable perfect forwarding. When the right-hand side of an assignment is an rvalue, then the left-hand side object can steal resources from the right-hand side object rather than performing a separate allocation, thus enabling move semantics.

The Visual C++ compiler in Visual Studio 2010 enables six C++0x core language features.

Perfect forwarding allows you to write a single function template that takes n arbitrary arguments and forwards them transparently to another arbitrary function. The nature of the argument (modifiable, const, lvalue or rvalue) is preserved in this forwarding process.

```
template <typename T1, typename T2> void functionA(T1&& t1, T2&& t2) {
  functionB(std::forward<T1>(t1), std::forward<T2>(t2));
}
```

A detailed explanation of rvalue references is out of scope for this article, so check the MSDN documentation at msdn.microsoft.com/library/dd293668(VS.100) for more information.

Static_assert allows testing assertions at compile time rather than at execution time. It lets you trigger compiler errors with custom error messages that are easy to read. Static_assert is especially useful for validating template parameters. For example, compiling the following code will give the error "error C2338: custom assert: n should be less than 5":

```
template <int n> struct StructA {
  static_assert(n < 5, "custom assert: n should be less than 5");
};

int _tmain(int argc, _TCHAR* argv[]) {
  StructA<4> s1;
  StructA<6> s2;
  return 0;
}
```

Nullptr adds type safety to null pointers and is closely related to rvalue references. The macro NULL (defined as 0) and literal 0 are commonly used as the null pointer. So far that has not been a problem, but they don't work very well in C++0x due to potential problems in perfect forwarding. So the nullptr keyword has been introduced particularly to avoid mysterious failures in perfect forwarding functions.

Nullptr is a constant of type nullptr_t, which is convertible to any pointer type, but not to other types like int or char. In addition to being used in perfect forwarding functions, nullptr can be used anywhere the macro NULL was used as a null pointer.

A note of caution, however: NULL is still supported by the compiler and has not yet been replaced by nullptr. This is mainly to avoid breaking existing code due to the pervasive and often inappropriate use of NULL. But in the future, nullptr should be used everywhere NULL was used, and NULL should be treated as a feature meant to support backward compatibility.

Finally, decltype allows the compiler to infer the return type of a function based on an arbitrary expression and makes perfect forwarding more generic. In past versions, for two arbitrary types T1 and T2, there was no way to deduce the type of an expression that used these two types. The decltype feature allows you to state, for example, an expression that has template arguments, such as sum<T1, T2>() has the type T1+T2.

> Rvalue references are a new reference type introduced in C++0x.

Standard Library Improvements

Substantial portions of the standard C++ library have been rewritten to take advantage of new C++0x language features and increase performance. In addition, many new algorithms have been introduced.

The standard library takes full advantage of rvalue references to improve performance. Types such as vector and list now have move constructors and move assignment operators of their own. Vector reallocations take advantage of move semantics by picking up move constructors, so if your types have move constructors and move assignment operators, the library picks that up automatically.

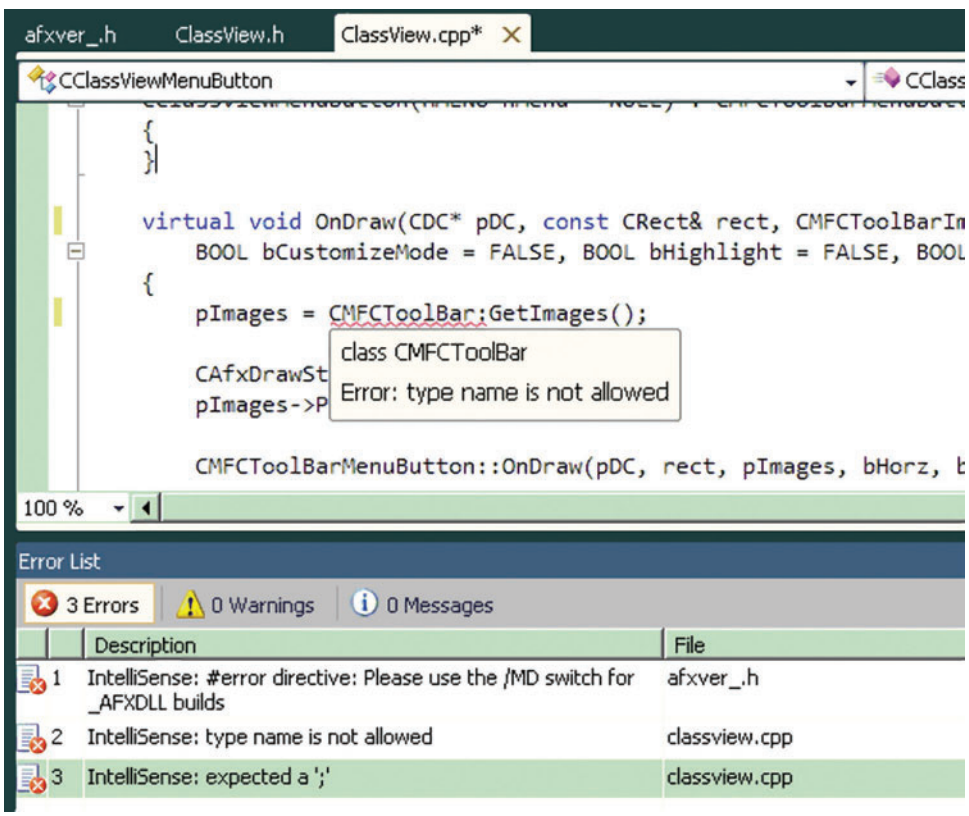


Figure 2 **Live Error Reporting Showing IntelliSense Errors**

You can now create a shared pointer to an object at the same time you are constructing the object with the help of the new C++0x function template make_shared<T>:

```
auto sp =
  make_shared<map<string,vector>>
  (args);
```

In Visual Studio 2008 you would have to write the following to get the same functionality:

```
shared_ptr<map<string,vector>>
  sp(new map<string,vector>(args));
```

Using make_shared<T> is more convenient (you'll have to type the type name fewer times), more robust (it avoids the classic unnamed shared_ptr leak because the pointer and the object are being created simultaneously), and more efficient (it performs one dynamic memory allocation instead of two).

The library now contains a new, safer smart pointer type, unique_ptr

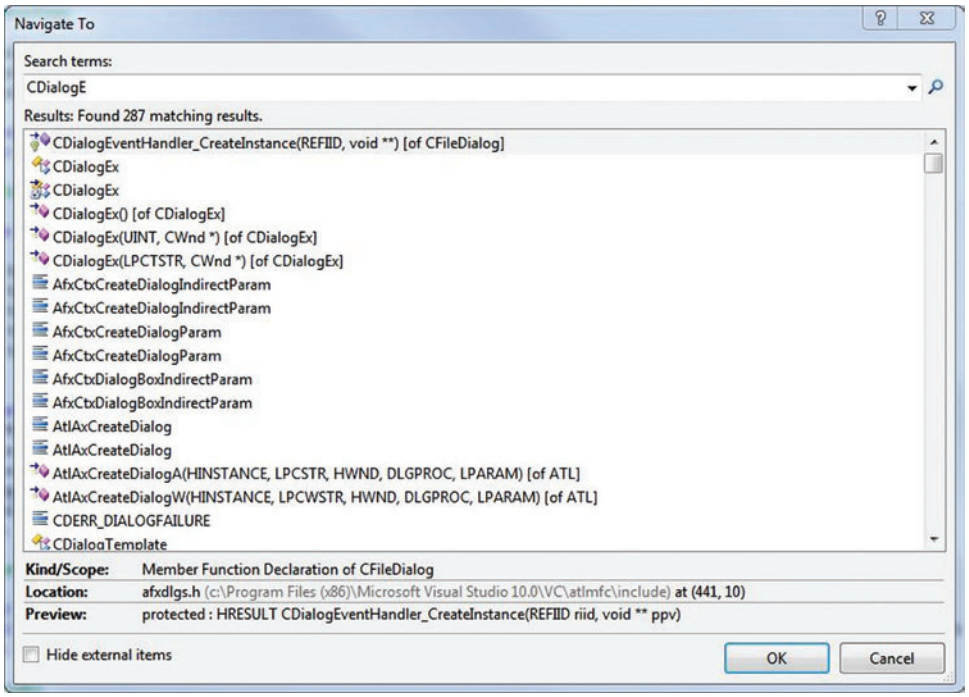


Figure 3 **Using the Navigate To Feature**

(which has been enabled by rvalue references). As a result, auto_ptr has been deprecated; unique_ptr avoids the pitfalls of auto_ptr by being movable, but not copyable. It allows you to implement strict ownership semantics without affecting safety. It also works well with Visual C++ 2010 containers that are aware of rvalue references.

Containers now have new member functions—cbegin and cend—that provide a way to use a const_iterator for inspection regardless of the type of container:

```
vector<int> v;

for (auto i = v.cbegin(); i != v.cend(); ++i) {
  // i is vector<int>::const_iterator
}
```

Visual Studio 2010 adds most of the algorithms proposed in various C++0x papers to the standard library. A subset of the Dinkumware conversions library is now available in the standard library, so now you can do conversions like UTF-8 to UTF-16 with ease. The standard library enables exception propagation via exception_ptr. Many updates have been made in the header <random>. There is a singly linked list named forward_list in this release. The library has a header <system_error> to improve diagnostics. Additionally, many of the TR1 features that existed in namespace std::tr1 in the previous release (like shared_ptr and regex) are now part of the standard library under the std namespace.

```
#ifdef FLAG
//Comments
void CPhotoViewerAppApp::OnFakeCrash()
{
    int *p = NULL;
    printf("%d",*p);
}
#endif
```

Figure 4 **Inactive Code Blocks Retain Colorization**

Concurrent Programming Improvements

Visual Studio 2010 introduces the Parallel Computing Platform, which helps you to write high-performance parallel code quickly while avoiding subtle concurrency bugs. This lets you dodge some of the classic problems relating to concurrency.

The Parallel Computing Platform has four major parts: the Concurrency Runtime (ConcRT), the Parallel Patterns Library (PPL), the Asynchronous Agents Library, and parallel debugging and profiling.

ConcRT is the lowest software layer that talks to the OS and arbitrates among multiple concurrent components competing for resources. Because it is a user mode process, it can reclaim resources when its cooperative blocking mechanisms are used. ConcRT is aware of locality and avoids switching tasks between different processors. It also employs Windows 7 User Mode Scheduling (UMS) so it can boost performance even when the cooperative blocking mechanism is not used.

> Visual Studio 2010 introduces the Parallel Computing Platform.

PPL supplies the patterns for writing parallel code. If a computation can be decomposed into sub-computations that can be represented by functions or function objects, each of these sub-computations can be represented by a task. The task concept is much closer to the problem domain, unlike threads that take you away from the problem domain by making you think about the hardware, OS, critical sections and so on. A task can execute concurrently with the other tasks independent of what the other tasks are doing. For example, sorting two different halves of an array can be done by two different tasks concurrently.

PPL includes parallel classes (task_handle, task_group and structured_task_group), parallel algorithms (parallel_invoke, parallel_for and parallel_for_each), parallel containers (combinable, concurrent_queue, and concurrent_vector), and ConcRT-aware synchronization primitives (critical_section, event and reader_writer_lock), all of which treat tasks as a first-class concept. All components of PPL live in the concurrency namespace.

Task groups allow you to execute a set of tasks and wait for them all to finish. So in the sort example, the tasks handling two halves of the array can make one task group. You are guaranteed that these

two tasks are completed at the end of the wait member function call, as shown in the code example of a recursive quicksort written using parallel tasks and lambdas:

```
void quicksort(vector<int>::iterator first,
vector<int>::iterator last) {
  if (last - first < 2) { return; }
  int pivot = *first;
  auto mid1 = partition(first, last, [=](int elem) {
    return elem < pivot; });
  auto mid2 = partition( mid1, last, [=](int elem) {
    return elem == pivot; });
  task_group g;
  g.run([=] { quicksort(first, mid1); });
  g.run([=] { quicksort(mid2, last); });
  g.wait();
}
```

A brand-new IntelliSense and browsing infrastructure is included in Visual Studio 2010.

This can be further improved by using a structured task group enabled by the parallel_invoke algorithm. It takes from two to 10 function objects and executes all of them in parallel using as many cores as ConcRT provides and waits for them to finish:

```
parallel_invoke(
  [=] { quicksort(first, mid1); },
  [=] { quicksort(mid2, last); } );

parallel_invoke(
  [=] { quicksort(first, mid1); },
  [=] { quicksort(mid2, last); } );
```

There could be multiple subtasks created by each of these tasks. The mapping between tasks and execution threads (and ensuring that all the cores are optimally utilized) is managed by ConcRT. So decomposing your computation into as many tasks as possible will help take advantage of all the available cores.

Another useful parallel algorithm is parallel_for, which can be used to iterate over indices in a concurrent fashion:

```
parallel_for(first, last, functor);
parallel_for(first, last, step, functor);
```

This concurrently calls function objects with each index, starting with first and ending with last.

The Asynchronous Agents Library gives you a dataflow-based programming model where computations are dependent on the required data becoming available. The library is based on the concepts of agents, message blocks and message-passing functions. An agent is a component of an application that does certain computations and communicates asynchronously with other agents to solve a bigger computation problem. This communication between agents is achieved via message-passing functions and message blocks.

Agents have an observable lifecycle that goes through various stages. They are not meant to be used for the fine-grained parallelism achieved by using PPL tasks. Agents are built on the scheduling and resource management components of ConcRT and help you avoid the issues that arise from the use of shared memory in concurrent applications.

You do not need to link against or redistribute any additional components to take advantage of these patterns. ConcRT, PPL and the Asynchronous Agents Library have been implemented within msvcr100.dll, msvcp100.dll and libcmt.lib/libcpmt.lib alongside the standard library. PPL and the Asynchronous Agents Library are mostly header-only implementations.

The Visual Studio debugger is now aware of ConcRT and makes it easy for you to debug concurrency issues—unlike Visual Studio 2008, which had no awareness of higher-level parallel concepts. Visual Studio 2010 has a concurrency profiler that allows you to visualize the behavior of parallel applications. The debugger has new windows that visualize the state of all tasks in an application and their call stacks. **Figure 1** shows the Parallel Tasks and Parallel Stacks windows.

MSBuild is now used to build C++ projects.

IntelliSense and Design-Time Productivity

A brand-new IntelliSense and browsing infrastructure is included in Visual Studio 2010. In addition to helping with scale and responsiveness on projects with large code bases, the infrastructure improvements have enabled some fresh design-time productivity features.

IntelliSense features like live error reporting and Quick Info tooltips are based on a new compiler front end, which parses the

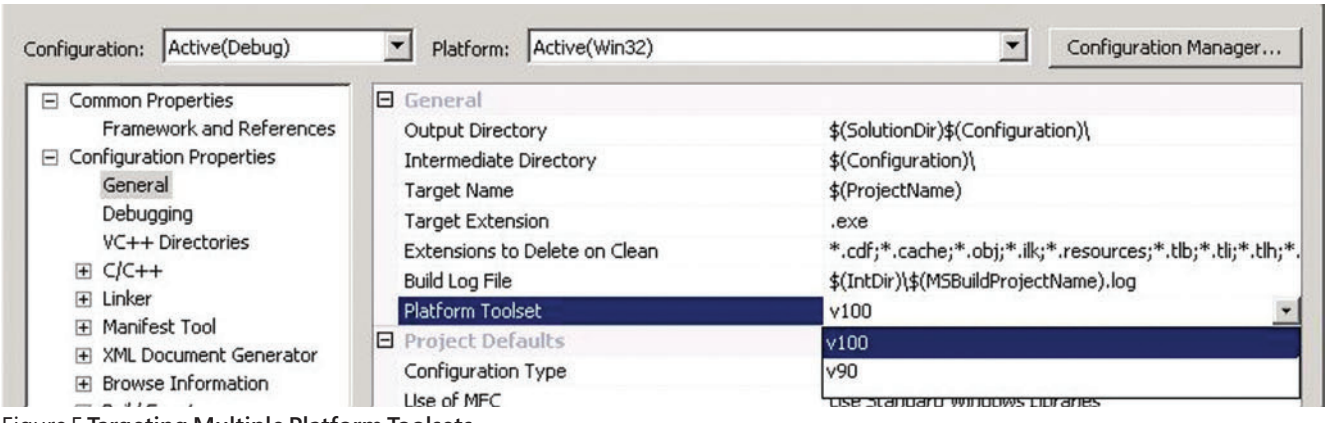


Figure 5 **Targeting Multiple Platform Toolsets**

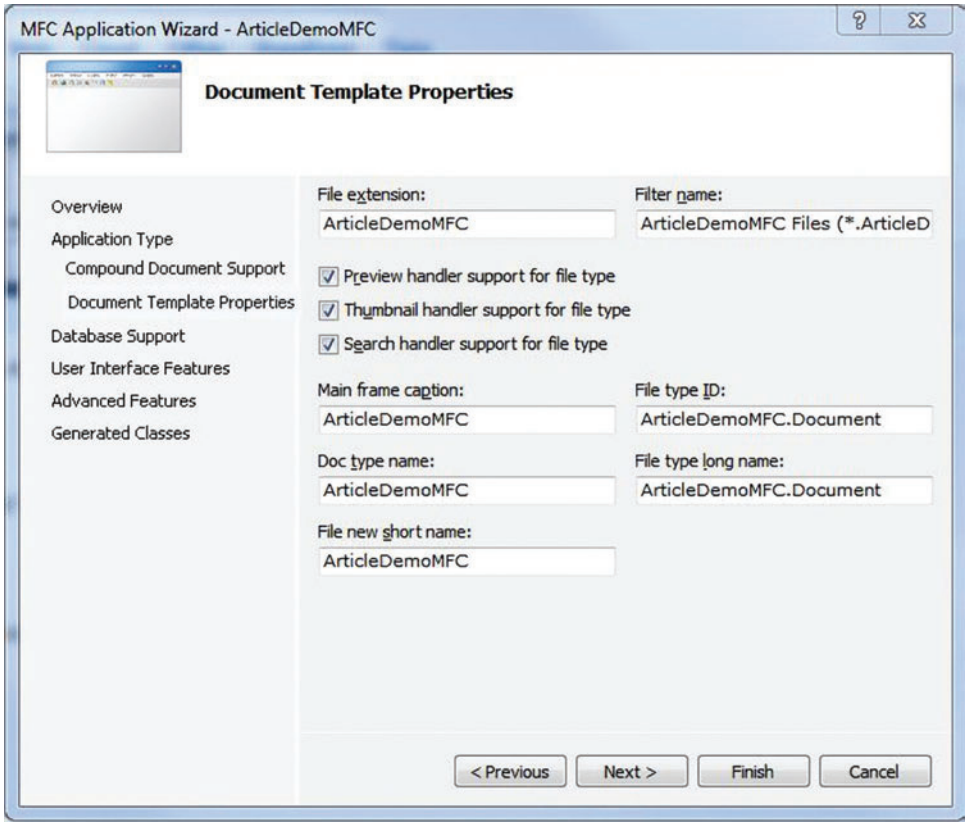


Figure 6 **MFC Application Wizard with File Handler Options**

full translation unit to provide rich and accurate information about code semantics, even while the code files are being modified.

All of the code-browsing features, like class view and class hierarchy, now use the source code information stored in a SQL database that enables indexing and has a fixed memory footprint. Unlike previous releases, the Visual Studio 2010 IDE is always responsive and you no longer have to wait while compilation units get reparsed in response to changes in a header file.

IntelliSense live error reporting (the familiar red squiggles) displays compiler-quality syntax and semantic errors during browsing and editing of code. Hovering the mouse over the error gives you the error message (see **Figure 2**). The error list window also shows the error from the file currently being viewed, as well as the IntelliSense errors from elsewhere in the compilation unit. All of this information is available to you without doing a build.

In addition, a list of relevant include files is displayed in a dropdown while typing #include, and the list refines as you type.

The new Navigate To (Edit | Navigate To or Ctrl+comma) feature will help you be more productive with file or symbol search. This feature gives you real-time search results, based on substrings as you type, matching your input strings for symbols and files across any project (see **Figure 3**). This feature also works for C# and Visual Basic files and is extensible.

Call Hierarchy (invoked using Ctrl+K, Ctrl+T or from the right-click menu) lets you navigate to all functions called from a particular function, and from all functions that make calls to a particular function. This is an improved version of the Call Browser feature that existed in previous versions of Visual Studio. The Call Hierarchy window is much better organized and provides both calls from and calls to trees for any function that appears in the same window.

Note that while all the code-browsing features are available for both pure C++ and C++/CLI, IntelliSense-related features like live error reporting and Quick Info will not be available for C++/CLI in the final release of Visual Studio 2010.

Other staple editor features are improved in this release, too. For example, the popular Find All References feature that is used to search for references to code elements (classes, class members, functions and so on) inside the entire solution is now more flexible. Search results can be further refined using a Resolve Results option from the right-click context menu.

Inactive code now retains semantic information by maintaining colorization (instead of becoming gray). **Figure 4** shows how the inactive code is dimmed but still shows different colors to convey the semantic information.

In addition to the features described already, the general editor experience is enhanced in Visual Studio 2010. The new Windows Presentation Foundation (WPF)-based IDE has been redesigned to remove clutter and improve readability. Document windows such as the code editor and Design view can now float outside the main IDE window and can be displayed on multiple monitors. It is easier to zoom in and out in the code editor window using the Ctrl key and the mouse wheel. The IDE also has improved support for extensibility.

MSBuild makes the C++ build system far more extensible.

Build and Project Systems

Visual Studio 2010 also boasts substantial improvements in the build system and the project system for C++ projects.

The most important change is that MSBuild is now used to build C++ projects. MSBuild is an extensible, XML-based build orchestration engine that has been used for C# and Visual Basic projects in previous versions of Visual Studio. MSBuild is now the common Microsoft build system for all languages. It can be used both in the build lab and on individual developer machines.

C++ build processes are now defined in terms of MSBuild target and task files and give you a greater degree of customizability, control and transparency.

The C++ project type has a new extension: .vcxproj. Visual Studio will automatically upgrade old .vcproj files and solutions to the new format. There is also a command-line tool, vcupgrade.exe, to upgrade single projects from the command line.

In the past, you could only use the toolset (compiler, libraries and so on) provided with your current version of Visual Studio. You had to wait until you were ready to migrate to the new toolset before you could start using the new IDE. Visual Studio 2010 solves that problem by allowing you to target multiple toolset versions to build against. For example, you could target the Visual C++ 9.0 compiler and libraries while working in Visual Studio 2010. **Figure 5** shows the native multi-targeting settings on the property page.

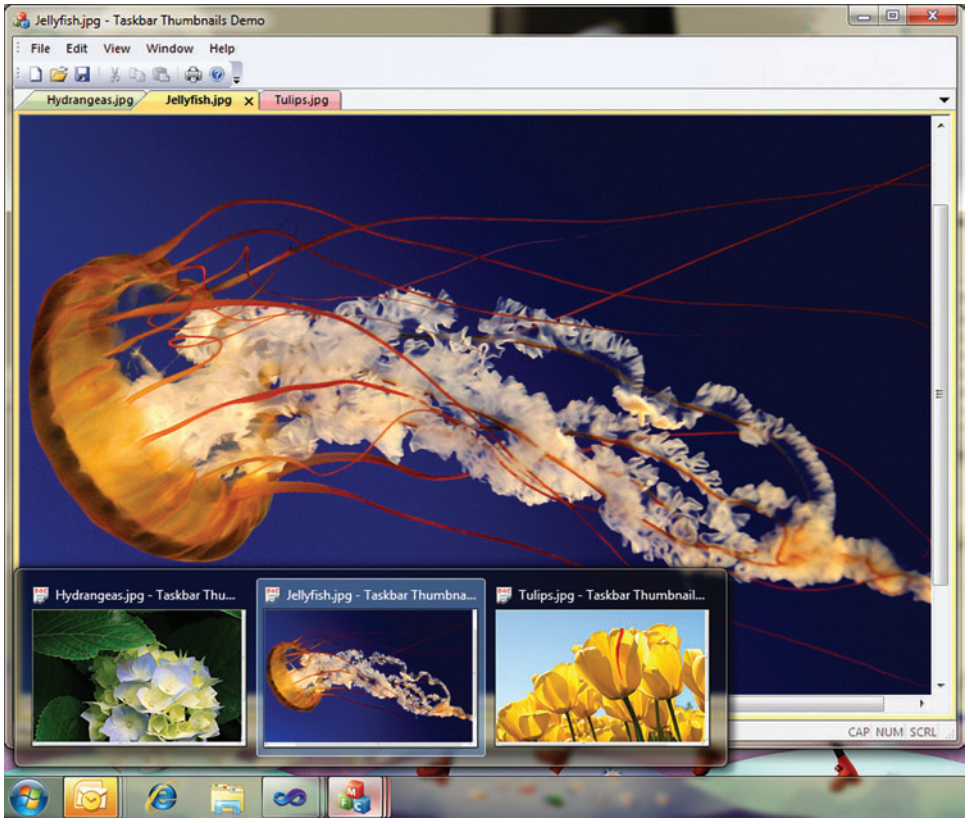


Figure 7 **Tabbed Thumbnail and Thumbnail Preview in an MFC Application**

Using MSBuild makes the C++ build system far more extensible. When the default build system is not sufficient to meet your needs, you can extend it by adding your own tool or any other build step. MSBuild uses tasks as reusable units of executable code to perform the build operations. You can create your own tasks and extend the build system by defining them in an XML file. MSBuild generates the tasks from these XML files on the fly.

Existing platforms and toolsets can be extended by adding .props and .targets files for additional steps into ImportBefore and ImportAfter folders. This is especially useful for providers of libraries and tools who would like to extend the existing build systems. You can also define your own platform toolset. Additionally, MSBuild provides better diagnostic information to make it easier for you to debug build problems, which also makes incremental builds more reliable. Plus, you can create build systems that are more closely tied to source control and the build lab and less dependent on developer machine configuration.

The project system that sits on top of the build system also takes advantage of the flexibility and extensibility provided by MSBuild. The project system understands the MSBuild processes and allows Visual Studio to transparently display information made available by MSBuild.

Customizations are visible and can be configured through the property pages. You can configure your project system to use your own platform (like the existing x86 or x64 platforms) or your own debugger. The property pages allow you to write and integrate components that dynamically update the value of properties that depend on context. The Visual Studio 2010 project system even allows you to write your own custom UI to read and write properties instead of using property pages.

Faster Compilation and Better Performance

In addition to the design-time experience improvements described so far, Visual Studio 2010 also improves the compilation speed, quality and performance for applications built with the Visual C++ compiler, as a result of multiple code generation enhancements to the compiler back end.

> Compilation speed on x64 platforms has been improved.

The performance of certain applications depends on the working set. The code size for the x64 architecture has been reduced in the range of 3 percent to 10 percent by making multiple optimizations in this release, resulting in a performance improvement for such applications.

Single Instruction Multiple Data (SIMD) code generation—which is important for game, audio, video and graphics developers—has been optimized for improved performance and code quality. Improvements include breaking false dependencies, vectorization of constant vector initializations, and better allocation of XMM registers to remove redundant loads, stores and moves. In addition, the __mm_set_**, __mm_setr_** and __mm_set1_** intrinsic family has been optimized.

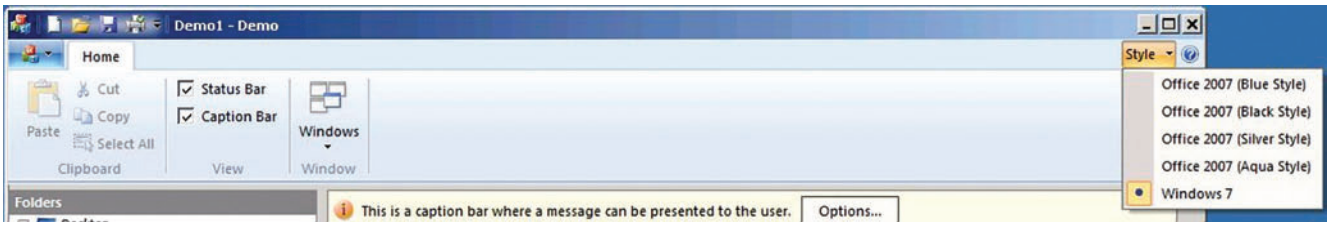


Figure 8 **Ribbon-Style Dropdown in an MFC Application**

For improved performance, applications should be built using Link Time Code Generation (LTCG) and Profile Guided Optimization (PGO).

Compilation speed on x64 platforms has been improved by making optimizations in x64 code generation. Compilation with LTCG, recommended for better optimization, usually takes longer than non-LTCG compilation especially in large applications. In Visual Studio 2010, the LTCG compilation has been improved by up to 30 percent. A dedicated thread to write PDB files has been introduced in this release, so you will see link time improvements when you use the /DEBUG switch.

PGO instrumentation runs have been made faster by adding support for no-lock versions of instrumented binaries. There is also a new POGO option, PogoSafeMode, that enables you to specify whether to use safe mode or fast mode when you optimize an application. Fast mode is the default behavior. Safe mode is thread-safe, but slower than fast mode.

The quality of compiler-generated code has been improved. There is now full support for Advanced Vector Extensions (AVX), which are very important for floating-point-intensive applications in AMD and Intel processors via intrinsic and /arch:AVX options. Floating-point computation is more precise with /fp:fast option.

Building Applications for Windows 7

Windows 7 introduced a number of exciting new technologies and features and added new APIs, and Visual Studio 2010 provides access to all the new Windows APIs. The Windows SDK components needed to write code for native Windows APIs are included in Visual Studio 2010. You can take advantage of innovations like Direct3D 11, DirectWrite, Direct2D and Windows Web Service APIs by using the SDK headers and libraries available in Visual Studio 2010.

In addition to making all the Windows APIs available to developers, this release of Visual Studio also makes it easier for you to write applications for Windows with the help of a beefed up MFC. You get access to substantial Windows 7 functionality through MFC libraries without having to write directly to native APIs. Your existing MFC applications will light up on Windows 7 just by recompiling. And your new applications can take full advantage of the new features.

> MFC now includes improved integration with the Windows shell.

MFC now includes improved integration with the Windows shell. Your application's integration with Windows Explorer can now be much better by making use of the file handlers for preview, thumbnails and search that have been added in this release. These features are provided as options in the MFC Application Wizard as shown in **Figure 6**. MFC will automatically generate the ATL DLL project that implements these handlers.

One of the most noticeable user interface changes in Windows 7 is the new taskbar. MFC allows you to quickly take advantage of features like jump lists, tabbed thumbnails, thumbnail preview,

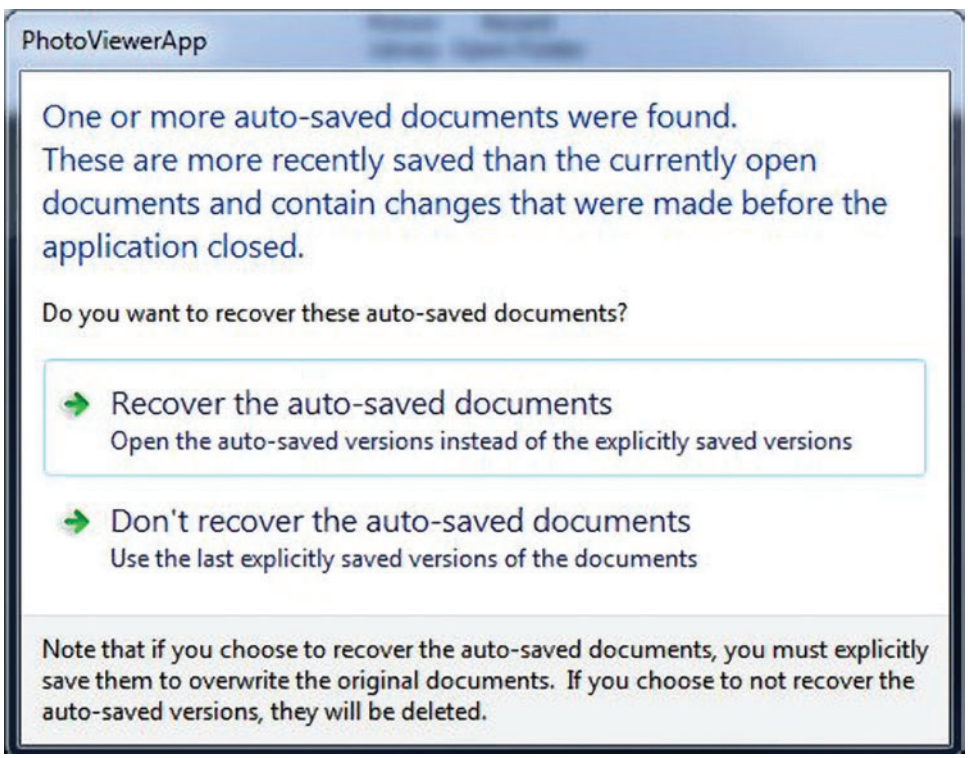


Figure 9 **Task Dialog**

progress bars, icon overlay and so on. **Figure 7** shows thumbnail previews and tabbed thumbnails for a tabbed MDI MFC application.

The ribbon UI now has a Windows 7-style ribbon, too, and your application can swap the UI on the fly any time during development from several Office-style ribbons to the Windows 7 ribbon by using the style dropdown as shown in **Figure 8**.

MFC enables your applications to become multi-touch aware and calls appropriate messages for you to handle when the various touch events occur. Just registering for touch and gesture events will route those events for your application. MFC also makes applications high-DPI-aware by default so they adapt to high-DPI screens and do not look pixelated or fuzzy. MFC internally scales and changes fonts and other elements to make sure your UI continues to look sharp on high DPI displays.

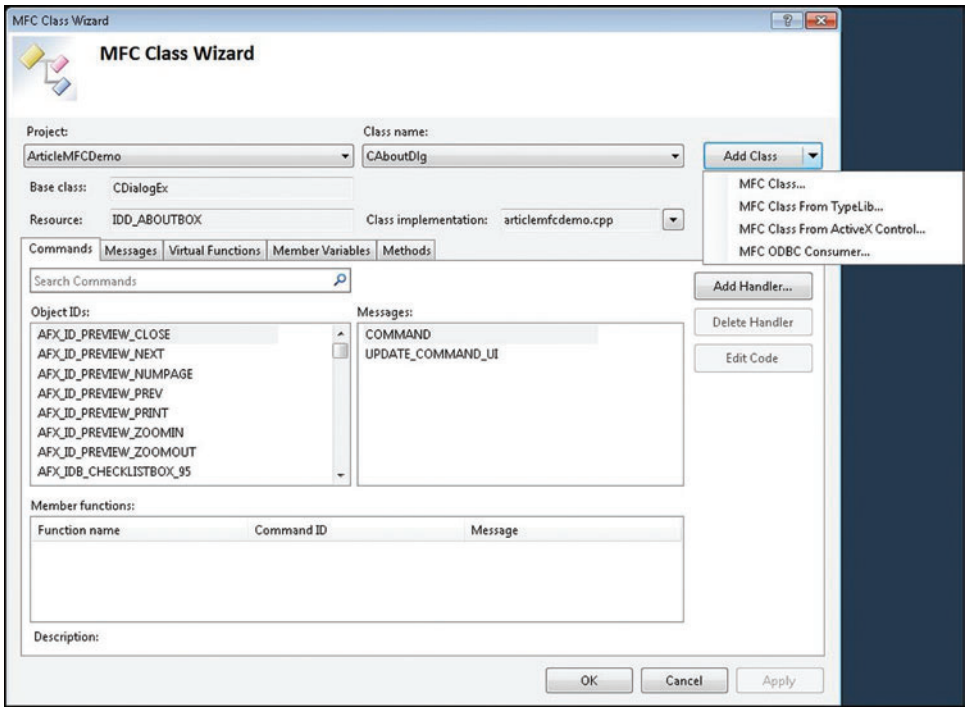


Figure 10 **MFC Class Wizard**

In addition to the new Windows 7 features, some other Windows features that have existed since Windows Vista but were not included in previous releases of MFC have been included now. For example, Restart Manager is a useful feature introduced in Windows Vista that enables an application to perform an application save before terminating. The application can invoke this feature and then restore its state when restarting. You can now take full advantage of Restart Manager in your MFC application to handle crashes and restarts more elegantly. Simply add a line of code to enable restart and recovery in your existing application:

```
CMyApp::CMyApp() {
  m_dwRestartManagerSupportFlags =
    AFX_RESTART_MANAGER_SUPPORT_RESTART;
// other lines of code ...
}
```

New MFC applications get this functionality automatically by using the MFC Application Wizard. The auto-save mechanism is available to applications that save documents, and the auto-save interval can be defined by the user. Applications can choose only restart support or application recover start (applicable to Doc/View type applications) in the MFC Application Wizard.

Another addition is the Windows Task dialog, which is an improved type of message box (see **Figure 9**). MFC now has a wrapper for the Task dialog that you can use in your applications.

MFC Class Wizard Is back

Not only has new functionality been added in the MFC library, this release also makes it easier to work with MFC in the Visual Studio IDE. One of the most commonly requested features, the MFC Class Wizard (shown in **Figure 10**), has been brought back and improved. You can now add classes, event handlers and other elements to your application using the MFC Class Wizard.

Another addition is the Ribbon Designer, which allows you to graphically design your ribbon UI (instead of defining it in code as in Visual Studio 2008) and store it as an XML resource. This designer is obviously useful for creating new applications, but existing applications can also take advantage of the designer to update their UIs. The XML definition can be created just by adding a line of code temporarily to the existing code definition of the ribbon UI:

```
m_wndRibbonBar.SaveToXMLFile(L"YourRibbon.mfcribbon-ms");
```

The MFC Class Wizard has been brought back and improved.

The resulting XML file can then be consumed as a resource file and further modifications can be made using the Ribbon Designer.

Wrapping Up

Visual Studio 2010 is a major release in the evolution of Visual C++ and makes lives of developers easier in many ways. I have barely scratched the surface of various C++-related improvements in this article. For further discussion on different features, please refer to the MSDN documentation and to the Visual C++ team's blog at blogs.msdn.com/vcblog, which was also used as the basis for some of the sections in this article. ■

SUMIT KUMAR *is a program manager in the Visual C++ IDE team. He holds an MS degree in Computer Science from the University of Texas at Dallas.*

THANKS to the following technical experts: *Stephan T. Lavavej, Marian Luparu and Tarek Madkour*

XCEED

DataGrid

for WPF

Serious recognition

Microsoft®
Visual Studio® Team System 2010

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

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

IBM®
U2 SystemBuilder™

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

Vincent Smith,
U2 Tools Product Manager at IBM

Microsoft Visual Studio
PARTNER

XCEED
MULTI-TALENTED COMPONENTS

Datagrids, transformed

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

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

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

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

Try it live on xceed.com

XCEED
MULTI-TALENTED COMPONENTS

Developing and Deploying Cloud Apps in Visual Studio 2010

Jim Nakashima, Hani Atassi and Danny Thorpe

There are many reasons to deploy an application or services onto Windows Azure, the Microsoft cloud services platform. These include reducing operation and hardware costs by paying for just what you use, building applications that are able to scale almost infinitely, enormous storage capacity, geo-location ... the list goes on and on.

Yet a platform is intellectually interesting only when developers can actually use it. Developers are the heart and soul of any platform release—the very definition of a successful release is the large number of developers deploying applications and services on it. Microsoft has always focused on providing the best development experience for a range of platforms—whether established or emerging—with Visual Studio, and that continues for cloud computing. Microsoft added direct support for building Windows Azure applications to Visual Studio 2010 and Visual Web Developer 2010 Express.

This article will walk you through using Visual Studio 2010 for the entirety of the Windows Azure application development lifecycle. Note that even if you aren't a Visual Studio user today, you can still evaluate Windows Azure development for free, using the Windows Azure support in Visual Web Developer 2010 Express.

This article is based on a prerelease version of Visual Studio 2010. All information is subject to change.

This article discusses:

- Developing a cloud service
- Hosting data in the cloud
- Debugging and deployment
- Promoting the hosted service

Technologies discussed:

Windows Azure, ASP.NET, Visual Studio 2010

Creating a Cloud Service

Start Visual Studio 2010, click on the File menu and choose New | Project to bring up the New Project dialog. Under Installed Templates | Visual C# (or Visual Basic), select the Cloud node. This displays an Enable Windows Azure Tools project template that, when clicked, will show you a page with a button to install the Windows Azure Tools for Visual Studio.

Before installing the Windows Azure Tools, be sure to install IIS on your machine. IIS is used by the local development simulation of the cloud. The easiest way to install IIS is by using the Web Platform Installer available at `microsoft.com/web`. Select the Platform tab and click to include the recommended products in the Web server.

Download and install the Windows Azure Tools and restart Visual Studio. As you'll see, the Enable Windows Azure Tools project template has been replaced by a Windows Azure Cloud Service project template. Select this template to bring up the New Cloud Service Project dialog shown in **Figure 1**. This dialog enables you to add roles to a cloud service.

A Windows Azure role is an individually scalable component running in the cloud where each instance of a role corresponds to a virtual machine (VM) instance.

There are two types of role:

- A Web role is a Web application running on IIS. It is accessible via an HTTP or HTTPS endpoint.
- A Worker role is a background processing application that runs arbitrary .NET code. It also has the ability to expose Internet-facing and internal endpoints.

As a practical example, I can have a Web role in my cloud service that implements a Web site my users can reach via a URL such as http://[somename].cloudapp.net. I can also have a Worker role that processes a set of data used by that Web role.

I can set the number of instances of each role independently, such as three Web role instances and two Worker role instances, and this corresponds to having three VMs in the cloud running my Web role and two VMs in the cloud running my Worker role.

You can use the New Cloud Service Project dialog to create a cloud service with any number of Web and Worker roles and use a different template for each role. You can choose which template to use to create each role. For example, you can create a Web role using the ASP.NET Web Role template, WCF Service Role template, or the ASP.NET MVC Role template.

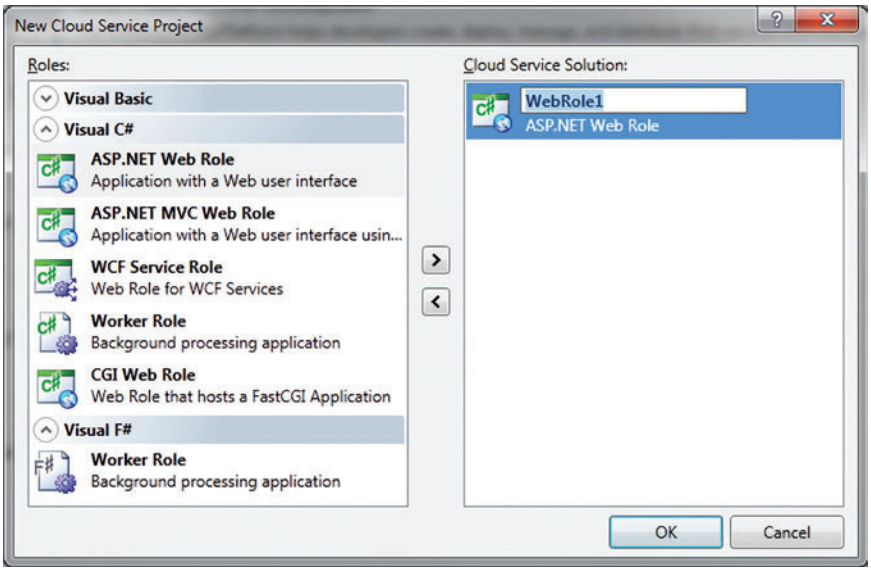


Figure 1 **Adding Roles to a New Cloud Service Project**

After adding roles to the cloud service and clicking OK, Visual Studio will create a solution that includes the cloud service project and a project corresponding to each role you added. **Figure 2** shows an example cloud service that contains two Web roles and a Worker role.

The Web roles are ASP.NET Web application projects with only a couple of differences. WebRole1 contains references to the following assemblies that are not referenced with a standard ASP.NET Web application:

- Microsoft.WindowsAzure.Diagnostics (diagnostics and logging APIs)
- Microsoft.WindowsAzure.ServiceRuntime (environment and runtime APIs)
- Microsoft.WindowsAzure.StorageClient (.NET API to access the Windows Azure storage services for blobs, tables and queues)

The file WebRole.cs contains code to set up logging and diagnostics and a trace listener in the web.config/app.config that allows you to use the standard .NET logging API.

The cloud service project acts as a deployment project, listing which roles are included in the cloud service, along with the definition and configuration files. It provides Windows Azure-specific run, debug and publish functionality.

It is easy to add or remove roles in the cloud service after project creation has completed. To add other roles to this cloud service, right-click on the Roles node in the cloud service and select Add | New Web Role Project or Add | New Worker Role Project. Selecting either of these options brings up the Add New Role dialog where you can choose which project template to use when adding the role.

You can add any ASP.NET Web Role project to the solution by right-clicking on the Roles node, selecting Add | Web Role Project in the solution, and selecting the project to associate as a Web role.

To delete, simply select the role to delete and hit the Delete key. The project can then be removed.

You can also right-click on the roles under the Roles node and select Properties to bring up a Configuration tab for that role (see **Figure 3**). This Configuration tab makes it easy to add or modify the values in both the ServiceConfiguration.cscfg and ServiceDefinition.csdef files.

When developing for Windows Azure, the cloud service project in your solution must be the StartUp project for debugging to work correctly. A project is the StartUp project when it is shown in bold in the Solution Explorer. To set the active project, right-click on the project and select Set as StartUp project.

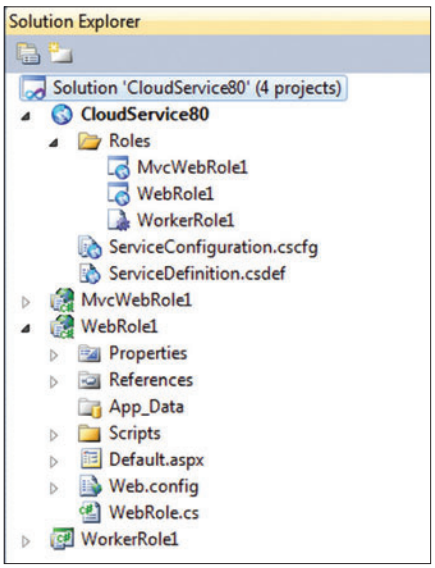


Figure 2 **Projects Created for Roles in the Cloud Service**

Data in the Cloud

Now that you have your solution set up for Windows Azure, you can leverage your ASP.NET skills to develop your application.

As you are coding, you'll want to consider the Windows Azure model for making your application scalable. To handle additional traffic to your application, you increase the number of instances for each role. This means requests will be load-balanced across your roles, and that will affect how you design and implement your application.

In particular, it dictates how you access and store your data. Many familiar data

storage and retrieval methods are not scalable, and therefore are not cloud-friendly. For example, storing data on the local file system shouldn't be used in the cloud because it doesn't scale.

To take advantage of the scaling nature of the cloud, you need to be aware of the new storage services. Windows Azure Storage provides scalable blob, queue, and table storage services, and Microsoft SQL Azure provides a cloud-based relational database service built on SQL Server technologies. Blobs are used for storage of named files along with metadata. The queue service provides reliable storage and delivery of messages. The table service gives you structured storage, where a table is a set of entities that each contain a set of properties.

To help developers use these services, the Windows Azure SDK ships with a Development Storage service that simulates the way these storage services run in the cloud. That is, developers can write their applications targeting the Development Storage services using the same APIs that target the cloud storage services.

Debugging

To demonstrate running and debugging on Windows Azure locally, let's use one of the samples from code.msdn.microsoft.com/windowsazuresamples. This MSDN Code Gallery page contains a number of code samples to help you get started with building scalable Web application and services that run on Windows Azure. Download the samples for Visual Studio 2010, then extract all the files to an accessible location like your Documents folder.

The Development Fabric requires running in elevated mode, so start Visual Studio 2010 as an administrator. Then, navigate to where you extracted the samples and open the Thumbnails solution, a sample service that demonstrates the use of a Web role and a Worker role, as well as the use of the StorageClient library to interact with both the Queue and Blob services.

When you open the solution, you'll notice three different projects. Thumbnails is the cloud service that associates two roles, Thumbnails_WebRole and Thumbnails_WorkerRole. Thumbnails_WebRole is the Web role project that provides a front-end application to the user to upload photos and adds a work item to the queue. Thumbnails_WorkerRole is the Worker role project that fetches the work item from the queue and creates thumbnails in the designated directory.

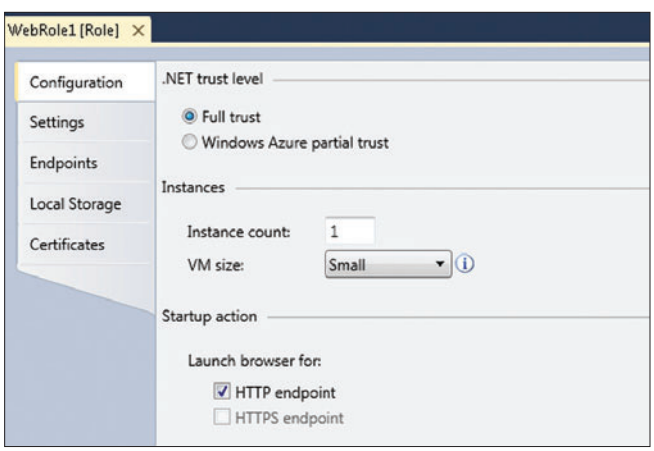


Figure 3 **Configuring a Role**

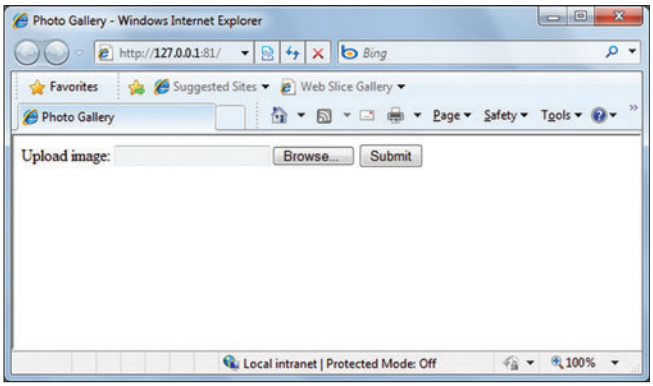


Figure 4 **Running the Thumbnails Sample**

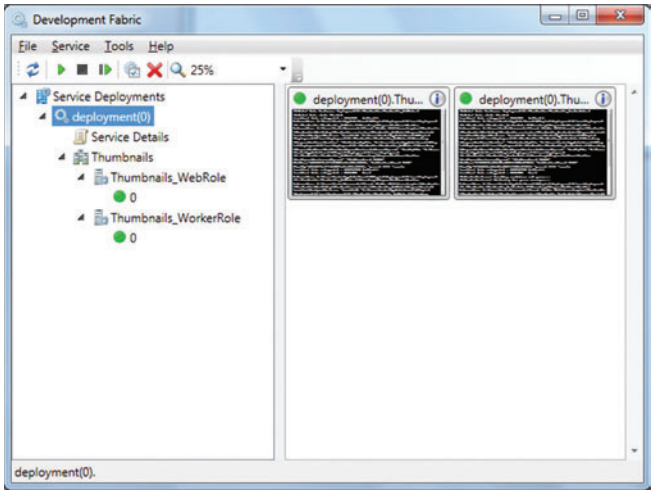


Figure 5 **The Development Fabric**

Add a breakpoint to the submitButton_Click method in the Default.aspx.cs file. This breakpoint will get hit when the user selects an image and clicks Submit on the page.

```
protected void submitButton_Click(
  object sender, EventArgs e) {
  if (upload.HasFile) {
    var name = string.Format("{0:10}", DateTime.Now.Ticks,
      Guid.NewGuid());
    GetPhotoGalleryContainer().GetBlockBlob(name).
UploadFromStream(upload.FileContent);
```

Now add a breakpoint in the Run method of the worker file, WorkerRole.cs, right after the code that tries to retrieve a message from the queue and checks if one actually exists. This breakpoint will get hit when the Web role puts a message in the queue that is retrieved by the worker.

```
while (true) {
  try {
    CloudQueueMessage msg = queue.GetMessage();
    if (msg != null) {
      string path = msg.AsString
```

To debug the application, go to the Debug menu and select Start Debugging. Visual Studio will build your project, start the Development Fabric, initialize the Development Storage (if run for the first time), package the deployment, attach to all role instances, and then launch the browser pointing to the Web role (see **Figure 4**).

At this point, you'll see that the browser points to your Web role and that the notifications area of the taskbar shows the Development Fabric has started. The Development Fabric is a simulation

environment that runs role instances on your machine in much the way they run in the real cloud.

Right-click on the Windows Azure notification icon in the taskbar and click on Show Development Fabric UI. This will launch the Development Fabric application itself, which allows you to perform various operations on your deployments, such as viewing logs and restarting and deleting deployments (see **Figure 5**). Notice that the Development Fabric contains a new deployment that hosts one Web role instance and one Worker role instance.

Look at the processes that Visual Studio attached to (Debug/Windows/Processes); you'll notice there are three: WaWebHost.exe, WaWorkerHost.exe and iexplore.exe.

WaWebHost (Windows Azure Web instance Host) and WaWorkerHost (Windows Azure Worker instance Host) host your Web role and Worker role instances, respectively. In the cloud, each instance is hosted in its own VM, whereas on the local development simulation each role instance is hosted in a separate process and Visual Studio attaches to all of them.

By default, Visual Studio attaches using the managed debugger. If you want to use another one, like the native debugger, pick it from the Properties of the corresponding role project. For Web role projects, the debugger option is located under the project properties Web tab. For Worker role projects, the option is under the project properties Debug tab.

By default, Visual Studio uses the script engine to attach to Internet Explorer. To debug Silverlight applications, you need to enable the Silverlight debugger from the Web role project Properties.

Browse to an image you'd like to upload and click Submit. Visual Studio stops at the breakpoint you set inside the submitButton_Click method, giving you all of the debugging features you'd expect from Visual Studio. Hit F5 to continue; the submitButton_Click method generates a unique name for the file, uploads the image stream to Blob storage, and adds a message on the queue that contains the file name.

Now you will see Visual Studio pause at the breakpoint set in the Worker role, which means the worker received a message from the queue and it is ready to process the image. Again, you have all of the debugging features you would expect.

Hit F5 to continue, the worker will get the file name from the message queue, retrieve the image stream from the Blob service, create a thumbnail image, and upload the new thumbnail image to the Blob service's thumbnails directory, which will be shown by the Web role.

Deployment

Now that you've created, edited and debugged your application locally, you're ready to deploy it to the cloud. Here's a good process to follow when deploying an application to Windows Azure:

- Get your application running locally in the Windows Azure Development Fabric using local storage.
- Run your application locally in the Development Fabric using a Windows Azure Storage Account.
- Run your application on Windows Azure using a Windows Azure Storage Account.

In the first stage, you can do all your development on your local machine using the Development Fabric and Development Storage as surrogates for the Windows Azure cloud infrastructure. You don't even need a network connection—you can develop and debug your Windows Azure application or service completely offline. You'll probably do 70 percent of your project development in this first stage.

In the second stage, you replace the local storage surrogate with the real deal, Windows Azure Storage, but retain the debugging and diagnostic advantages of executing your Windows Azure application code in the local Development Fabric. You can set breakpoints in source code, step through source code line by line, evaluate expressions, and examine call stacks while your Windows Azure app interacts with cloud storage. You'll probably spend 20 percent to 25 percent of your project cycle in this stage, refining your code and testing against real-world asynchronous operations.

By the time you get to the third stage, your Windows Azure application should be all but finished. You should be feature-complete and code-complete except for bug fixes. Your main focus in this stage is testing and performance tuning. With your Windows Azure application executing in the cloud, you won't have the luxury of source code debugging so you'll have to fall back to diagnostic logging.

You probably don't want your brand-new Windows Azure application to be seen by the whole wide world on your target URL the instant you upload it to the Windows Azure cloud. The Windows Azure Hosted Service has the notion of multiple deployments within the same hosted service. Each Windows Azure Hosted Service has a private staging deployment area where you can quietly test your code in the cloud, and a public production deployment area where you release your tested code for your customers to use.

Staging deployments can only be accessed via a unique URL that is prefixed by a GUID assigned by the system. Production deployments can be the target of custom domain name mappings for easy access.

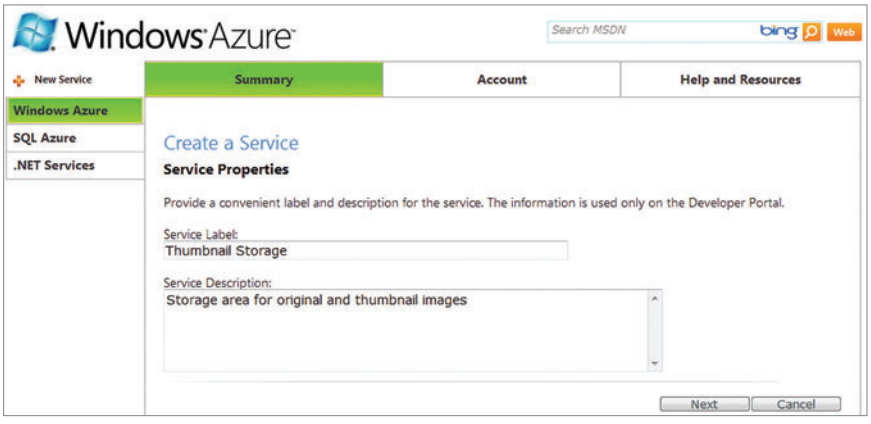


Figure 6 **Configuring a Storage Service**

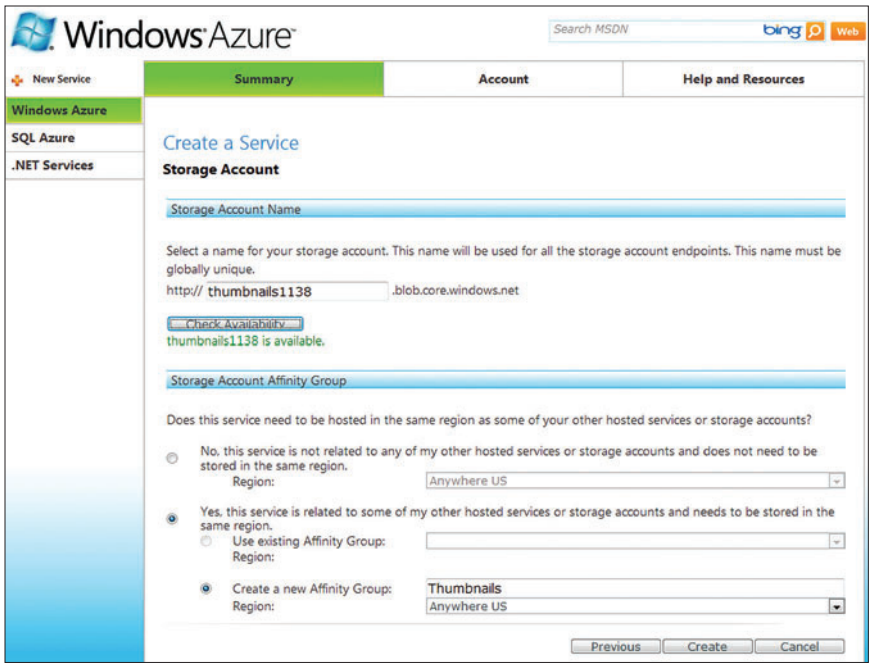


Figure 7 **Configuring a Storage Domain**

Moving to Windows Azure Storage

Now that the Thumbnails service is running on a local machine in the Developer Fabric, let's upgrade it to work against a Windows Azure Storage account in the cloud. This involves getting a storage account and making the configuration changes to run the Thumbnails service against your storage account. Executing locally but using cloud storage for data is a great way to ensure that your code will run when hosted on Windows Azure.

Start by navigating to the Windows Azure Developer Portal (windows.azure.com) and sign in with your Live ID. From the page, select New Service and click on Storage Account. In the Create a Service page (see **Figure 6**), fill in a friendly name for the storage account (this is not the service/domain name; you will be prompted for that on the next page). Optionally enter a project description. Then fill in the service/domain name on the next page (see **Figure 7**). This domain name is global so you may need to fiddle with it a bit to get a name that is not already in use by another service. Be sure to click on Check Availability to test for uniqueness.

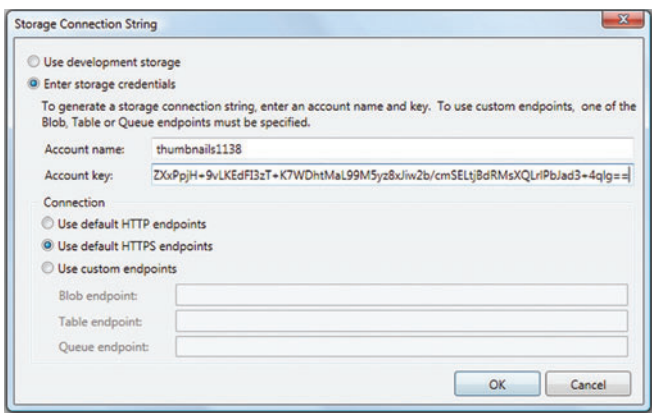


Figure 8 **Storage Connection String Properties**

It's a good idea to create an affinity group to ensure that your storage and the hosted services that use it are located in the same datacenter whenever possible. Affinity groups can also specify a preference for a geographical region so that service and storage are as close to your target audience as possible to minimize network transit time.

Click Create and you'll see a summary page for your new storage account.

Now you need to tell your Thumbnails service to use your new storage account to store its thumbnails and images. Double-click on the Thumbnails_WebRole node under Roles in the Visual Studio Solution Explorer to open its properties page. Select the Settings tab, select DataConnectionString, and click on the edit button on the far right of the grid row. This brings up the Storage Connection String dialog, shown in **Figure 8**.

"Use development storage" is the default for new Azure projects. Click on "Enter storage credentials" and then enter your storage service details. In the Account name field, type the service/domain name you entered when creating your storage service. This is the first part of your domain, thumbnails1138 (all lower case) in the example.

In the Account key field, enter the Primary Access Key displayed on your storage service summary Web page. You can select the key text in the browser, copy it, and paste it into the AccountSharedKey edit box.

Note that the Secondary Access Key provides the same access to your storage account as the Primary Access Key and is generated as a backup should your Primary Access Key be compromised. Regenerating a key invalidates the old key, shutting down storage access to anyone still using the old key.

Leave the connection endpoints set to use the default HTTPS endpoints and click OK.

Repeat these steps for the Thumbnails_WorkerRole under Roles as well, so that the Web role will be talking to the same storage service as the Worker role. When you have multiple roles to update, you can copy the connection string value to the clipboard and just paste it into the DataConnectionString value cell in each of the other role properties pages to save a little time.

Once you have switched the Web and Worker roles to use Windows Azure storage, hit F5 in Visual Studio to debug your cloud service and ensure that everything works correctly.

You'll see that the Web page URL you're debugging is still locally hosted, but the URLs of the thumbnails (visible in their Properties dialogs) now point to Windows Azure Storage.

Note that this stage of development—local services with cloud storage—will most likely have the worst performance of the three stages of deployment. Stage 1 (local/local) will be pretty snappy because everything is in the same box and has a total audience of one (you!). Stage 3 (cloud/cloud) will have the benefit of cloud scale hardware and datacenter affinity. Stage 2 (local/cloud),

Ken Getz

Robert Green

Don Kiely

Chris Menegay

Moving to Visual Studio 2010? Start off on the right foot by learning about new features and functionality from AppDev. With our award-winning online learning, you will...

- Learn directly from Microsoft MVPs and experts
- See how things are done with practical, step-by-step demos
- Practice new skills with hands-on lab exercises and sample code
- Use comprehensive printable courseware for handy reference
- Ask questions of our instructors via our AppDev Edge member site

BEST VALUE! Get our 85-course online learning library covering Visual Studio®, SQL Server®, SharePoint® and more. You'll get our first three Visual Studio 2010 courses now, plus future courses for the following technologies included FREE!

FREE

Visual Basic® 2010 • Visual C#® 2010 • ASP.NET 4.0
Windows Presentation Foundation • MVC 2 • Silverlight™ 4

We offer cost-effective learning solutions for individuals and teams of all sizes.

Visit our Web site or call us to get started today!

www.appdev.com • 1-800-578-2062

+44(0)1903 740 624 (Europe/UK)

AppDev
Technical learning...in a class by itself®

however, will be executing code that reasonably assumes its storage is nearby, but in fact the data is probably many network hops away from the local execution environment.

Don't evaluate the performance of your app in the local/cloud configuration. You can turn that network lag to your advantage, though: use this stage as a stress test to see how your code handles exaggerated response times from its storage service. If you have handled something with a synchronous call that should really be asynchronous, you'll probably notice it pretty quickly.

Once you have everything working in the local/cloud configuration, you are ready to deploy your code to a Windows Azure hosted service.

Creating a Hosted Service

To create the fully hosted service, go back to the Windows Azure Developer Portal and sign in with your Live ID. Click on New Service, then on Hosted Services. This takes you to a page where you can specify the project-friendly name and description used by the developer portal. Enter a service label and optionally a service description. Click Next. Now you can enter a domain name for your hosted service project (see **Figure 9**). Set the affinity group to match the Thumbnails affinity group created earlier with the storage service. After the project is created, you will see a summary page for your project.

To deploy the project from Visual Studio, right-click on the Thumbnails project node in Solution Explorer and select Publish. This command builds a package of your Windows Azure application binaries and related files, opens Windows Explorer to show the local directory where the package was built, and launches your default Web browser to browse to the developer portal.

From the developer portal Web page, navigate to the Thumbnails Hosted Service summary page and click Deploy from under Staging to bring up the Staging Deployment page (see **Figure 10**). This is where you specify the package and the configuration file to upload.

You can copy the path from the Windows Explorer window that Visual Studio opened into this File Open dialog, which makes it easy for you to select the Service Package (.cspkg) and Service Configuration files.

The portal will then upload the package and deploy your cloud service to staging, which puts your roles in the Allocated state. This means your Windows Azure application has been provisioned with datacenter hardware, but it is not yet scheduled to run.

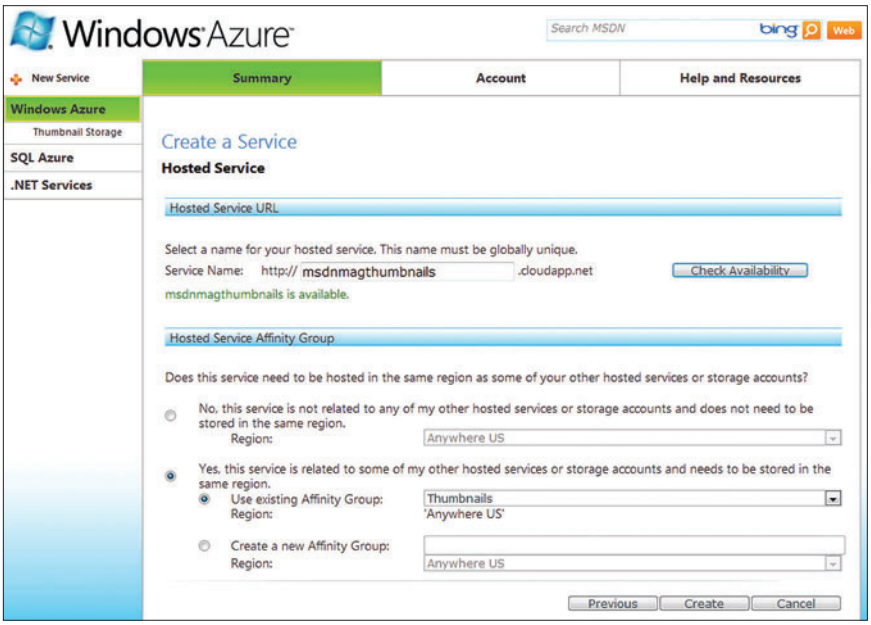


Figure 9 **Configuring a Hosted Service**

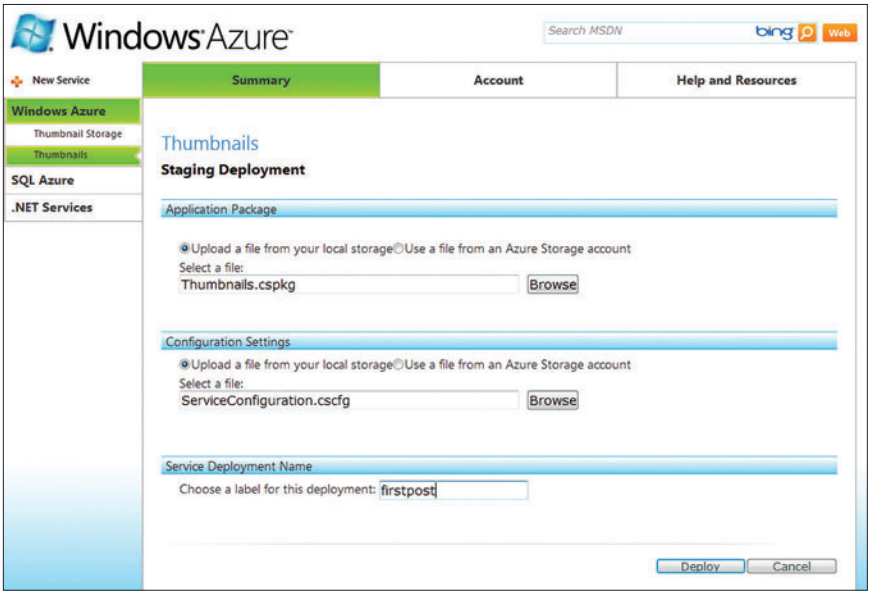


Figure 10 **Choosing the Package to Deploy**

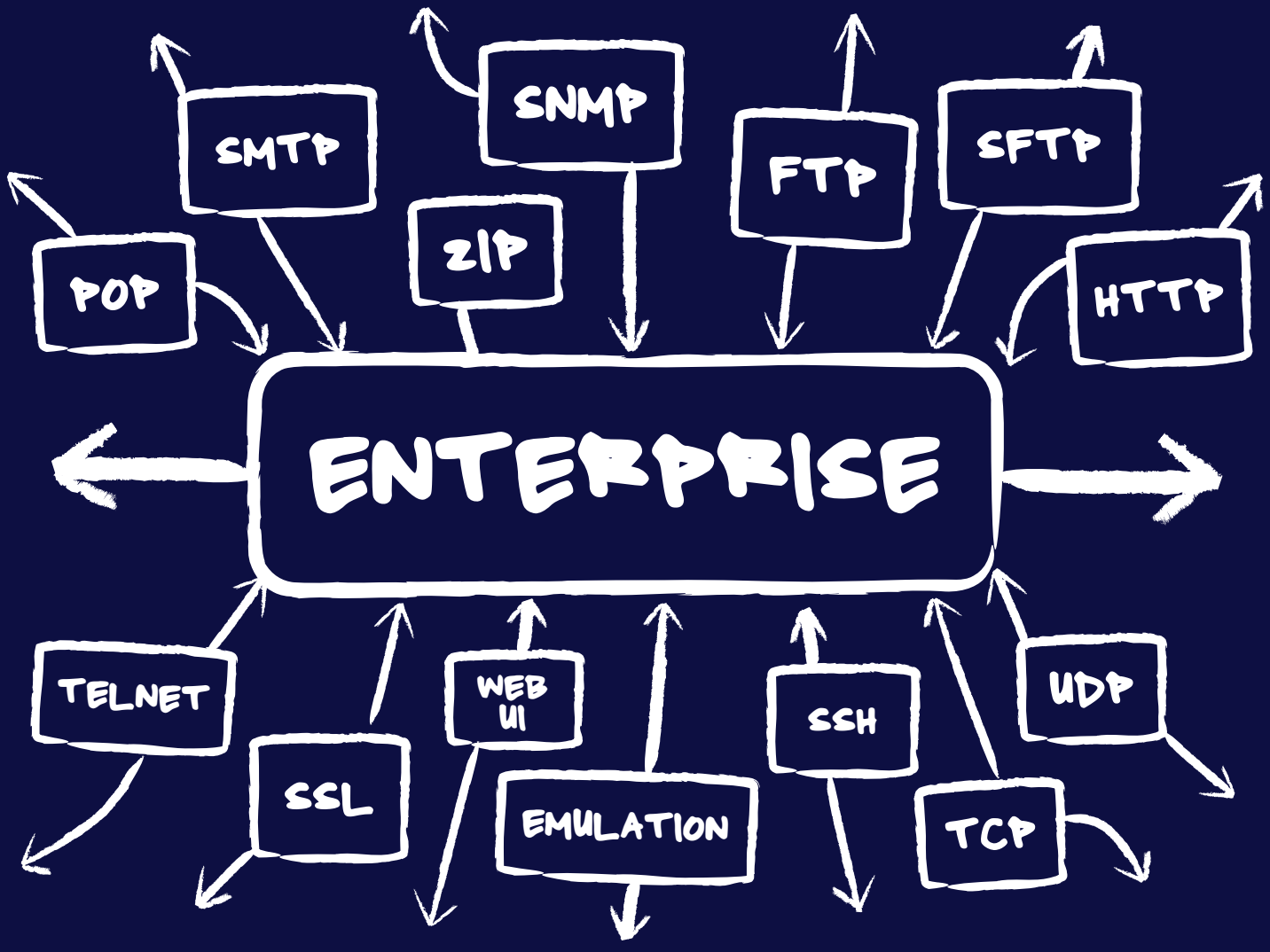
To test your Cloud Service in the Staging area, you need to run it. Click Run. This will put your Web role in the Initializing state. When the Web role is ready, the state will change to Started.

Note that there are plans to make deploying straight to the cloud from Visual Studio even simpler in future updates, but this procedure will remain valid even when those features are in place.

Once your roles have been started, you can test by navigating to the staging URL (the cloudapp.net URL that begins with a GUID).

Provisioning an application in the datacenter involves a lot of heavy lifting behind the scenes. Allow at least 10 minutes for your app to transition from Initializing to Started. Now aren't you glad you have the Developer Fabric on your local machine?

After you're happy with your cloud service on staging, you can promote it to production by clicking on the button shown in **Figure 11**.



Internet Connectivity for the Enterprise

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

PowerSNMP for ActiveX and .NET

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

PowerWEB for ASP.NET

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

PowerTCP for ActiveX and .NET

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

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

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

Download a fully functional product trial today!

DART.com

When the details of your deployment show up on the Production side of the page, your service will be up and running at its final URL.

If you need to test a hotfix for your app, or just want to continue development in parallel to the running production deployment, you can upload a new package to staging and test it via the staging URL. Promoting to production is actually a swap: staging moves to production and what was in production goes into staging.

Figure 11 Promotion Button

While you can directly upload to production, it is highly recommended to always deploy first to staging and perform some degree of acceptance testing before pushing it to production.

Note that multiple deployments will default to using the same storage service—if you need data isolation between production and staging (for example, so staging can wipe its database without affecting production data), you will need to modify your storage bindings for each deployment before you deploy them. This is typically done by using two sets of storage and migrating production data to staging before promoting staging to production.

Updating the Service Configuration

If you want to spin up additional instances of your service in anticipation of additional load, or shut down unused instances, you can do that by modifying the deployment configuration on the fly. You don't have to redeploy the entire package, just the service configuration file. In the portal, click Configure to update the service configuration (cscfg), either by uploading a new service configuration file created by Visual Studio or by using the editor provided by the developer portal.

Let's now add an HTTPS endpoint to the Thumbnails application. This is a three-step process. You need to configure the endpoint, configure the certificate and upload the certificate.

To configure the endpoint, open up the configuration UI on the Web role by right-clicking on the Thumbnails_WebRole node under the Roles node in Solution Explorer and selecting Properties. Switch to the Endpoints tab and click the checkbox to select HTTPS. This adds the HTTPS endpoint, but doesn't specify the certificate.

Switch to the Configuration page and uncheck the "Launch browser for: HTTP endpoint" option. By unselecting this option, on run or debug of the cloud service, the default browser will be launched only for the HTTPS endpoint.

In Visual Studio, click Debug | Start Debugging to package and run the cloud service on the local development simulation. The development simulation always uses a self-signed certificate issued to and issued by 127.0.0.1, which corresponds to the local host. Because the certificate is not trusted, the Web browser will come up with a certificate error. This is expected. Click "Continue to this website (not recommended)" to browse to the Web site.

To make the certificate trusted and therefore not see the certificate errors, you can install the certificate to the Trusted Root Certification Authorities certificate store. Do so only if you have an appropriate understanding of the security implications.

To configure a certificate for use in the cloud, you need a certificate that will be uploaded to the cloud and to configure that certificate for the role. For the purpose of this article, we'll create and use a self-signed certificate. Create a self-signed certificate by opening the IIS Manager, selecting Server Certificates, and clicking Create Self-Signed Certificate under the Actions heading on the far right of the dialog. After creating the certificate, click Export to export the certificate to a .pfx file.

Navigate to the Windows Azure Developer Portal and select the Hosted Service component to deploy to. Under the Certificates heading, select Manage. Upload the certificate by entering the name of the .pfx file and the corresponding password you entered during the export step. Copy the thumbprint of the certificate after it is installed to your hosted service component.

To configure the certificate, go back to Visual Studio, open the Thumbnails_WebRole configuration UI, click the Certificates tab and click Add Certificate. Give the certificate a name (for example, sslCert), paste in the thumbprint, and leave the store location and name at the default of LocalMachine and My.

The certificates configuration page allows you to specify for a given role what certificates should be installed to the VM instances for that role and in which stores to install those certificates. In other words, this same process can be used for any certificates you want to have on your VMs in the cloud, and not just for SSL certificates.

Finally, switch to the Endpoints tab and select sslCert for the HTTPS certificate.

Now deploy your application. You will now be able to access your Web site via HTTP and HTTPS. Because we uploaded a self-signed certificate, the browser will display a certificate error when browsing to your HTTPS endpoint. Using a real signed certificate will solve this problem.

Wrapping Up

The Windows Azure Tools and Visual Studio 2010 make it easy to create, edit, configure, debug and deploy applications that run on Windows Azure. They allow you to leverage your existing skills with ASP.NET and with Visual Studio.

The Windows Azure Tools add-in is designed for both Visual Studio 2010 and Visual Studio 2008. The easiest way to install the Windows Azure Tools for Visual Studio 2008 is by using the Web Platform Installer available at microsoft.com/web. Be sure to add the developer tools scenarios in the options.

For the latest news and information about Windows Azure, please see windowsazure.com and blogs.msdn.com/jnak. ■

Hani Atassi *is a software engineer on the Windows Azure Tools team. Prior to his work on cloud tools, Atassi spent time developing Windows Vista and Microsoft.com.*

Danny Thorpe *is a principal software engineer on the Windows Azure Tools team. In past lives he was a contributing founder of Google Gears at Google and a Delphi Compiler Architect at Borland.*

Jim Nakashima *is a program manager on the Windows Azure Tools team focusing on building end-to-end developer experiences for Windows Azure. Nakashima spent a number of years working on the Windows Presentation Foundation and Silverlight Designer and tools before being attracted to the infinite possibilities of the cloud.*

Thanks to the following technical experts for reviewing this article: *Anson Horton and Gus Perez*

BEST SELLER

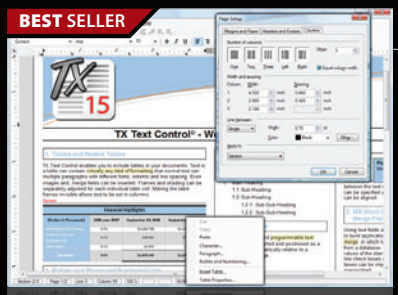
ContourCube | from $900.00

Contour Components

OLAP component for interactive reporting and data analysis.

- Embed Business Intelligence functionality into database applications
- Zero report coding - design reports with drag and drop
- Self-service interactive reporting - get hundreds of reports by managing rows/columns
- Royalty free - only development licenses are needed
- Provides extremely fast processing of large data volumes

BEST SELLER



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

TX Text Control word processing components

Word processing components for Visual Studio .NET.

- Add professional word processing to your applications
- Royalty-free Windows Forms text box
- True WYSIWYG, nested tables, text frames, headers and footers, images, bullets, structured numbered lists, zoom, dialog boxes, section breaks, page columns
- Load, save and edit DOCX, DOC, PDF, PDF/A, RTF, HTML, TXT and XML

BEST SELLER

FusionCharts | from $195.02

InfoSoft Global
empowering human thoughts

Interactive and animated charts for ASP and ASP.NET apps.

- Liven up your Web applications using animated Flash charts
- Create AJAX-enabled charts that can change at client-side without invoking server requests
- Export charts as images/PDF and data as CSV for use in reporting
- Also create gauges, financial charts, Gantt charts, funnel charts and over 550 maps
- Used by over 15,000 customers and 250,000 users in 110 countries

BEST SELLER

Janus WinForms Controls Suite | from $757.44

Janus systems

Add Outlook style interfaces to your WinForms applications.

- Janus GridEX for .NET (Outlook style grid)
- Janus Schedule for .NET and Timeline for .NET (Outlook style calendar view and journal)
- Janus ButtonBar for .NET and ExplorerBar for .NET (Outlook style shortcut bars)
- Janus UI Bars and Ribbon Control (menus, toolbars, panels, tab controls and Office 2007 ribbon)
- Now includes Office 2007 visual style for all controls

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

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

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

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

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

GSA Schedule Contract GS-35F-0188R

Microsoft®
Visual Studio® LAUNCH
conference and expo

Be a part of the Microsoft launch for Visual Studio 2010!

CO-LOCATED WITH:

MICROSOFT ASP.NET & SILVERLIGHT

SQL SERVER MAGAZINE

3 Conferences for the Price of One!

EXCITING KEYNOTES DELIVERED BY ...

BOB MUGLIA
MICROSOFT
President of the Server and Tools Business (STB)

SCOTT GUTHRIE
MICROSOFT
Corporate Vice President, .NET Developer Platform

Keep your competitive edge as you dive deep into Visual Studio 2010, Silverlight 4, ASP.NET 4.0, Microsoft SQL Server, AJAX and MVC.

- Attend Microsoft executive launch keynotes
- Train with Microsoft architects and industry experts
- Join the launch and party with colleagues from around the world
- Visit the exciting expo hall and meet our partners

Book your room with us at 800.438.6720 for the $149/night rate at the Bellagio. (space limited)

CHECK WEB SITE FOR DESCRIPTIONS OF SESSIONS AND WORKSHOPS

www.DevConnections.com • 800.438.6720 • 203.400.6121 • Register Today!

Microsoft SQL Server Magazine ineta CODE Magazine

Powered by Microsoft and DevConnections Magazine

APRIL 12-14, 2010 • LAS VEGAS, NV

BELLAGIO HOTEL AND CASINO

A SAMPLING OF THE MANY IN-DEPTH SESSIONS

VISUAL STUDIO

- Design Better Systems with Microsoft Visual Studio 2010
- Improve the Management of Your Team's Code Using Visual Studio Team Foundation Server 2010
- Develop for SharePoint 2010
- Build Better User Interfaces with WPF and Silverlight Technologies in Visual Studio 2010
- Use Windows Workflow Foundation to Encapsulate Business Processes
- Learn the New Features of Visual Basic 10 and Visual C# 4.0

ASP.NET

- Build Client Side User Interfaces with ASP.NET 4.0 AJAX
- Watch a Complete End-to-End Solution Being Built using .NET on the Microsoft Web Platform
- Learn the New Features for Web Forms in Visual Studio 2010
- Deploy Your Applications with MSDeploy and Visual Studio 2010
- Explore All of ASP.NET MVC
- Optimize Performance of Your ASP.NET Applications
- Learn How to Use Silverlight 4

SQL SERVER

- Design, Build and Deploy an Analysis Services Database
- Walk Through Microsoft SQL Server 2008 R2 StreamInsight
- Learn About Spatial Types and Methods Supported in SQL Server 2008
- Deliver Self-Serve BI with Gemini
- Explore the New Reporting Services
- Encryption and Debugging
- Learn How to Use Master Data Services in SQL Server 2008 R2
- Achieve SQL Server High Availability with Virtualization
- Combine Disparate Data Sources in SSRS Reports
- Learn to Deploy SQL Server Azure Databases

A SAMPLING OF INDUSTRY EXPERTS :

RICHARD CAMPBELL
STRANGELOOP NETWORKS

ROCKY LHOTKA
MAGENIC

KATHLEEN DOLLARD
APPVENTURE

DOUG SEVEN
MICROSOFT

KIMBERLY L. TRIPP
SQLSKILLS.COM

JUVAL LOWY
IDESIGN, INC.

BILLY HOLLIS
AUTHOR

MICHELE LEROUX BUSTAMANTE
IDESIGN, INC.

.NET Rocks! DevConnections magazine TECH Conferences Inc. PENTON MEDIA THE CODE PROJECT Your Development Resource

Entity Framework 4.0 and WCF Data Services 4.0 in Visual Studio 2010

Elisa Flasko

Among its many new improvements, Visual Studio 2010 introduces the much-anticipated Entity Framework 4.0 and WCF Data Services 4.0 (formerly ADO.NET Data Services), which together simplify how you model, consume and produce data.

This article discusses prerelease versions of Visual Studio 2010 and ADO.NET Data Services Update for Silverlight 3 CTP 3. All information is subject to change.

This article discusses:

- Foreign key support in Entity Framework 4.0
- Model first with Entity Framework 4.0
- Custom code generation
- Using POCO entities
- Lazy loading
- Creating a WCF Data Service
- Consuming a WCF Data Service in Silverlight
- Data binding in WCF Data Services 4.0
- Server-driven paging

Technologies discussed:

Visual Studio 2010, ADO.NET Data Services Update for Silverlight 3 CTP 3, Entity Framework 4.0, WCF Data Services 4.0

Code download available at:

code.msdn.microsoft.com/mag201004VSData

Entity Framework 4.0 (EF 4.0) focuses on enabling and simplifying two primary scenarios: domain-centric application development and the traditional data-centric "forms over data." It introduces features such as model first development, which allows you to create a model and have customized T-SQL generated for you; support for persistence ignorance; foreign keys; lazy loading and custom code generation for entities.

WCF Data Services 4.0 focuses on enabling updates to the Open Data Protocol (odata.org) and on its new features, including two-way data binding for Windows Presentation Foundation (WPF) and Silverlight, row count, server-driven paging, enhanced binary large object support and support for projections.

Using a sample weblog application (MyBlog), I will explore the new features in both EF and WCF Data Services and explain how the technologies work together to simplify the way data is modeled and consumed. This sample application will have both an ASP.NET Web application that provides a read-only view of blog posts, and a Silverlight blog administrator client that allows the blog owner to edit posts. I will begin the application using model first to create an Entity Data Model (EDM), and then generate a database and the code to interact with that database. This sample will also make use of the ADO.NET Data Services Update for Silverlight 3 CTP 3.

Getting Started with EF 4.0

I'll begin with the ASP.NET Web Application project. (My application is called BlogModel; you can download the accompanying code at

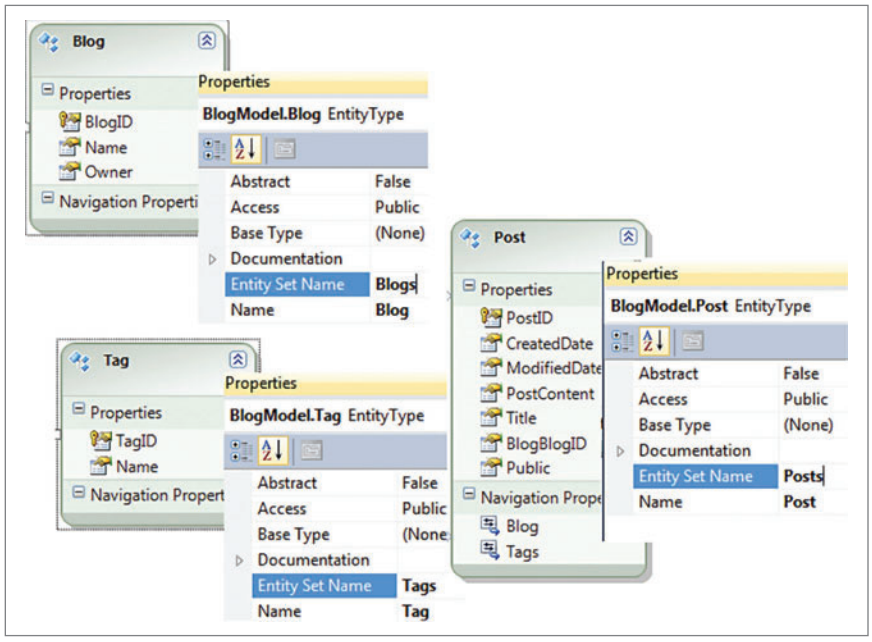


Figure 1 **Blog, Post and Tag Entities and Associated Property Settings**

code.msdn.microsoft.com/mag201004VSData.) To get started with EF, I use the Add New Item wizard to add an ADO.NET EDM, and select an Empty model that I'll call BlogModel as well. By right-clicking on the empty Designer surface and choosing Properties, you can see the default Entity Container name, BlogModelContainer, in this case. First I'll change the name to BlogContext, and then I'll create the model.

> Entity Framework 4.0 focuses on enabling and simplifying two primary scenarios: domain-centric application development and the traditional data-centric "forms over data."

MyBlog requires three Entities I'll call Blog, Post and Tag, as shown in **Figure 1**. To create them, I drag an Entity from the toolbox to the design surface, then right-click and select Properties to edit the entity properties. I'll also need a few scalar properties on each of the entities (right-click on the entity and select Add | Scalar Property).

Foreign Key Support in EF 4.0

Next, I'll add relationships between these entities. Right-click on the design surface and select Add | Association, as shown in **Figure 2**. EF now supports foreign keys, allowing the inclusion of foreign key properties on an entity. Notice that adding the relationship added a BlogBlogID property (the foreign key) to the Post entity.

The inclusion of foreign key properties on entities simplifies a number of key coding patterns, including data binding, dynamic data, concurrency control and *n*-tier development. For example, if I'm databinding a grid that shows products and I have the CategoryID (a foreign key value) in a grid but I don't have the corresponding Category object, the foreign key support in EF means I no longer need to take on the cost of a query to separately pull back the Category object.

Model First with EF 4.0

Now that the model is built (see **Figure 3**), the application needs a database. In this case, MyBlog is a new application and doesn't yet have a database. I don't want to create the database myself; I'd rather just have it done for me—and I can. With model first in EF 4.0, Visual Studio can now generate not only custom code for the entities, but also a database based on the model just created.

First, I need to create the empty database to which I will apply the generated schema. To do this, I open Server Explorer, right-click on the Data Connections node and select Create New SQL Server Database (see **Figure 4**). With the empty database created, I right-click on the model design surface and select Generate Database from Model. Walking through the Generate Database wizard creates a BlogModel.edmx.sql file. With this new file open, it's simple to right-click on the file and execute the SQL script to create the schema for my database.

Custom Code Generation with EF 4.0

At this point, there are a number of next steps possible, one of which is to customize the code that EF generates, using T4 Templates. In Visual Studio 2008 SP1, although EF provided some hooks

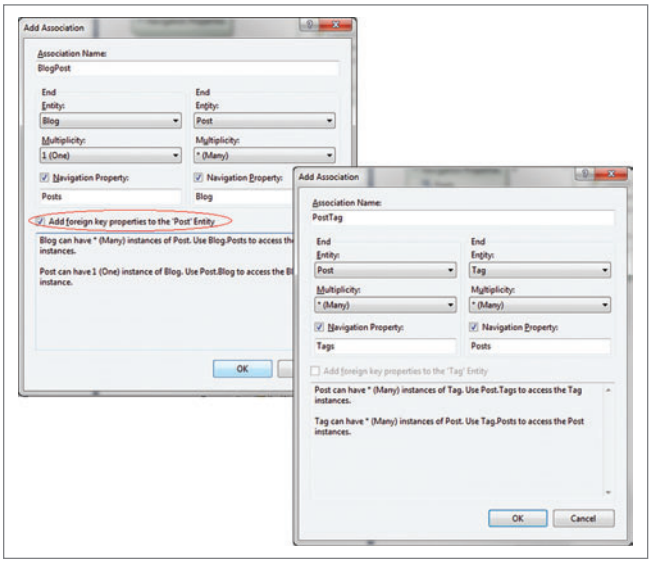


Figure 2 **Associations Between Blog, Post and Tag Entities**

for customizing code generation, it was relatively inflexible and hard to use. EF 4.0 now leverages T4 Templates, providing a much simpler, more flexible and more powerful way to customize generated code.

To add a new T4 Template to the project, right-click on the Entity Designer surface and select Add Code Generation Item. From here, choose any currently installed template to use as a starting point, or view the templates available in the Online Gallery. To use the default EF template as the starting point for this project, I'll choose the ADO.NET EntityObject Generator template; by default the template is called Model1.tt. By adding a code generation template in this manner, EF automatically disables default code generation for the model. The generated code is removed from BlogModel.Designer.cs and now exists in Model1.cs. At this point, the template can be edited to customize the entities it will generate, and every time the .tt file is saved, the dependent code will be regenerated. For more information on editing and using T4 Templates with EF 4.0, check out the ADO.NET Team blog at blogs.msdn.com/adonet.

Using POCO Entities with EF 4.0

Visual Studio 2008 SP1 imposed a number of restrictions on entity classes that made it difficult, to say the least, to build classes that were truly independent of persistence concerns. One of the most requested features in EF 4.0 is the ability to create Plain Old CLR Object (POCO) types for entities that work with EF and don't impose the type of restrictions in Visual Studio 2008 SP1.

> Entity Framework now supports foreign keys, allowing the inclusion of foreign key properties on an entity.

Let's go back to the MyBlog sample. I'm going to create POCO objects for the three entities—Blog, Post and Tag. First, code generation needs to be turned off and I need to remove the .tt file I added in the last section. To check out the properties of the model, right-click on the Entity Designer surface. As shown in **Figure 5**, there is a property named Code Generation Strategy that needs to be set to None to turn off code generation.

Note that this property is set to None automatically when you add a Code Generation Item (T4 Template) to a project. If the project currently has a .tt file included, you will need to remove it before using POCO objects. From here, the classes for the POCO objects can be added—Blog.cs, Post.cs and Tag.cs, as shown in **Figures 6**, **7** and **8**.

Last, I need to create the context class, which is much like the ObjectContext implementation generated with default code generation, but I'll call it BlogContext. It will inherit from the ObjectContext class. The context is the class that is persistence-aware. It will allow the composition of queries, materialize entities and save changes back to the database (see **Figure 9**).

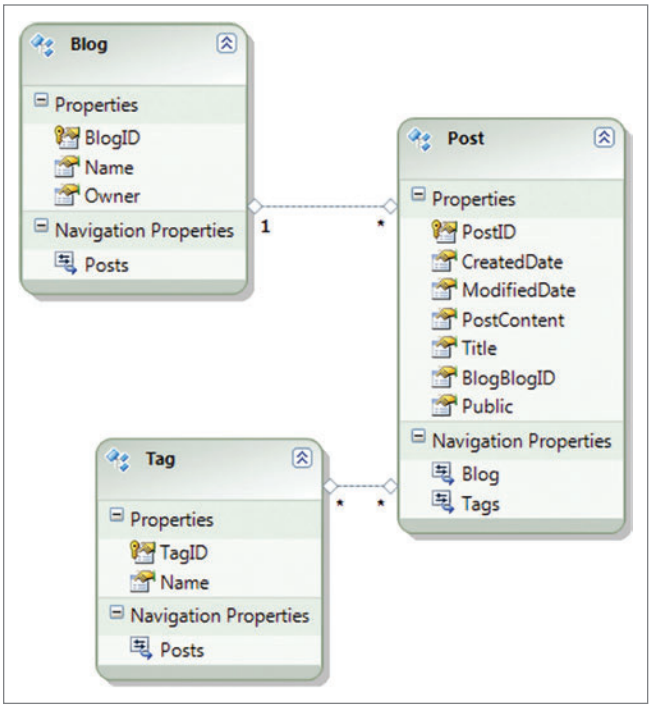


Figure 3 **Blog Model**

Lazy Loading

In Visual Studio 2008 SP1, EF supported two basic ways of loading related entities, both ensuring that the application hits the database only when explicitly told to do so, using either the Load method to explicitly load the related entities, or the Include method to eagerly load related entities within a query. Another of the most requested features now in EF 4.0 is lazy loading. When performing explicit loads is not required, you can use lazy loading (also known as deferred loading) to load related entities when a navigation property is accessed for the first time. In Visual Studio 2010, this is done by making the navigation properties virtual properties.

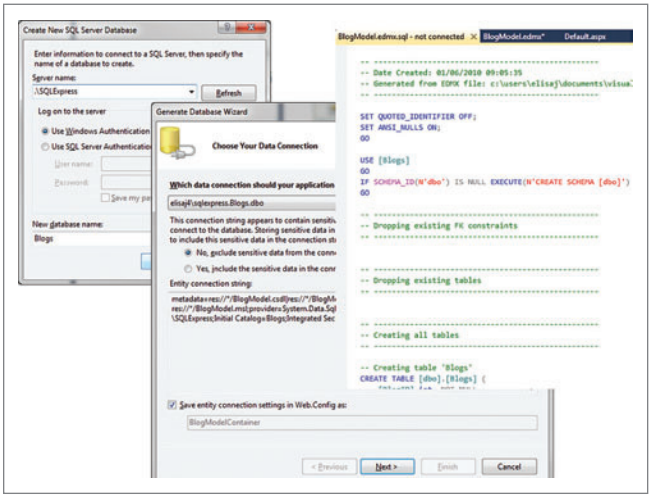


Figure 4 **Create a New Empty Database and Generate Database Schema from EDM**

In the MyBlog sample, the public List<Post> Posts property in both Blog.cs and Tag.cs would become public virtual List<Post> Posts, and the public List<Tag> Tags property in Post.cs would become public virtual List<Tag> Tags. EF would then create a proxy type at runtime that knows how to perform a load so that no additional code changes are necessary. However, because the MyBlog sample uses WCF Data Services to expose entities over an Open Data Protocol (OData) service, the application doesn't make use of lazy loading.

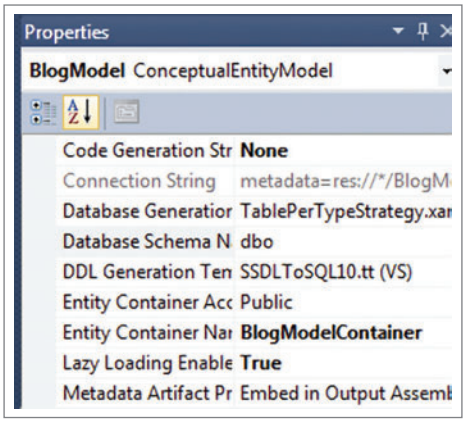


Figure 5 **Code Generation Strategy Property**

Creating a WCF Data Service in Visual Studio 2010

MyBlog takes advantage of the near-turnkey solution supplied by WCF Data Services to provide an OData service over an EDM, and includes a Silverlight blog administrator client that uses the OData service. The Open Data Protocol is a data-sharing standard that breaks down silos and fosters a powerful, interoperative ecosystem for data consumers (clients) and producers (services), enabling more applications to make sense of a broader set of data.

With an EDM and database set up, adding a new WCF Data Service to the application is simple; using the Add New Item wizard, I add a WCF Data Service (I'll call it BlogService). This generates a BlogService.svc file that represents the skeleton of the service and is pointed at the EDM by providing it with the context created earlier. Because the service is fully locked down by default, access to the entity sets that are to be made available over the service must be explicitly allowed using config.SetEntitySetAccessRule. To do this, a rule is set for each EntitySet that is made available, as seen in **Figure 10**.

(Note: If you download the sample code for this article, you'll notice it uses a very simple Forms Authentication scheme to secure the site, and the remaining examples will also use this scheme to filter data based on the currently logged-in user. Because implementing Forms Authentication is beyond the scope of this article, however, I'm going to skip the details here.)

With the service up and running, the next step is to filter the results based on the user currently logged in so that only Blogs owned by that user are returned. You can accomplished this by adding Query Interceptors as seen in **Figure 11** to restrict the entities returned by a query.

Consuming a WCF Data Service in Silverlight

The details of building a Silverlight UI are beyond the scope of this article, so I'll gloss over some of them. But before digging in to how to hook up the data service to a Silverlight app, I'll add a new Silverlight application to the project that contains the default Silverlight page MainPage.xaml. To that I add a basic DataGrid, ComboBox, Button and a couple of labels. With a skeleton Silverlight app ready (see **Figure 12**), we can hook up the data service.

To start, the Silverlight application needs objects that represent each of the entities defined by the data service. To do this, you use the Add Service Reference wizard in Visual Studio to automatically generate client classes for the data service. (Note that to use Add Service Reference, I need to temporarily disable the authorization checks implemented on the service so that Add Service Reference has full access to the service. I point the Add Service Reference wizard at the base URI for the service, which in MyBlog is localhost:48009/BlogService.svc).

Data Binding in WCF Data Services 4.0

Improved support for data binding in WCF Data Services 4.0 adds a new collection type, DataServiceCollection, to the client library, extending ObservableCollection. However, in Silverlight 3, data binding is turned off by default when a service reference is added to the project. So to take advantage of the new data binding

Figure 6 **POCO Object for Blog Entity**

```
public class Blog
{
  public intBlogID
  {
    get;
    set;
  }
  public string Name
  {
    get;
    set;
  }
  public string Owner
  {
    get;
    set;
  }
  public List<Post> Posts
  {
    get { return _posts; }
    set { _posts = value; }
  }
  List<Post> _posts = new List<Post>();
}
```

Figure 7 **POCO Object for Tag Entity**

```
public class Tag
{
  public int TagID
  {
    get;
    set;
  }
  public string Name
  {
    get;
    set;
  }
  public List<Post> Posts
  {
    get { return _posts; }
    set { _posts = value; }
  }
  List<Post> _posts = new List<Post>();
}
```

Figure 8 **POCO Object for Post Entity**

```
public class Post
{
  public int PostID
  {
    get;
    set;
  }
  public DateTime CreatedDate
  {
    get;
    set;
  }
  public DateTime ModifiedDate
  {
    get;
    set;
  }
  public string Title
  {
    get;
    set;
  }

  public string PostContent
  {
    get;
    set;
  }
  public Blog Blog
  {
    get;
    set;
  }
  public int BlogBlogID
  {
    get;
    set;
  }
  public Boolean Public
  {
    get;
    set;
  }
  public List<Tag> Tags
  {
    get { return _tags; }
    set { _tags = value; }
  }
  private List<Tag> _tags = new List<Tag>();
}
```

Figure 9 **BlogContext**

```
public class BlogContext : ObjectContext
{
  public BlogContext()
    : base("name=BlogContext", "BlogContext")
      {
      }
  public ObjectSet<Blog> Blogs
  {
    get
    {
      if (_Blogs == null)
      {
        _Blogs =
          base.CreateObjectSet<Blog>("Blogs");
      }
      return _Blogs;
    }
  }
  private ObjectSet<Blog> _Blogs;
  public ObjectSet<Post> Posts
  {
    get
    {
      if (_Posts == null)
      {
        _Posts =
          base.CreateObjectSet<Post>("Posts");
      }
    return _Posts;
    }
  }
  private ObjectSet<Post> _Posts;
  public ObjectSet<Tag> Tags
  {
    get
    {
      if (_Tags == null)
    {
      _Tags = base.CreateObjectSet<Tag>("Tags");
    }
    return _Tags;
    }
  }
  private ObjectSet<Tag> _Tags;
}
```

functionality in WCF Data Services, data binding needs to be turned on and the service reference needs to be updated. From the Solution Explorer, click the Show All Files button and expand the BlogService item under the Service References node. Double-click the Reference.datasvc map file and replace the Parameters element with the XML snippet shown here:

```
<Parameters>
  <Parameter Name="UseDataServiceCollection" Value="true" />
  <Parameter Name="Version" Value="2.0" />
</Parameters>
```

Setting the UseDataServiceCollection parameter to true auto-generates client-side types that implement the INotifyPropertyChanged and INotifyCollectionChanged interfaces. This means that any changes made to contents of a DataServiceCollection or the entities in the collection are reflected on the client context. It also means that if an entity in the collection is requeried, any changes to that entity are reflected in the entities in the DataServiceCollection. And it means that because the DataServiceCollection implements the standard binding interfaces, it can be bound as the DataSource to most WPF and Silverlight controls.

Figure 10 **BlogService.svc**

```
public class BlogService : DataService<BlogContext>
{
    // This method is called only once to initialize service-wide
policies.
    public static void InitializeService(DataServiceConfiguration config)
    {
       // TODO: set rules to indicate which entity sets and service
       // operations are visible, updatable, etc.
       // Examples:
       config.SetEntitySetAccessRule("Blogs", EntitySetRights.All);
       config.SetEntitySetAccessRule("Posts", EntitySetRights.All);
       config.SetEntitySetAccessRule("Tags", EntitySetRights.All);
       // config.SetServiceOperationAccessRule("MyServiceOperation",
       // ServiceOperationRights.All);
       config.DataServiceBehavior.MaxProtocolVersion =
       DataServiceProtocolVersion.V2;
    }
}
```

Going back to the MyBlog sample, the next step is to create a connection to the service by creating a new DataServiceContext and use it to query the service. **Figure 13** includes MainPage.xaml and MainPage.xaml.cs and shows the creation of a new DataServiceContext, querying the service for all blogs—in this case the service returns all blogs owned by the logged-in user—and binding the blogs to a ComboBox on the Silverlight application.

To bind the DataGrid, a cboBlogs_SelectionChanged() method is added:

```
private void cboBlogs_SelectionChanged(object sender,
SelectionChangedEventArgs e)
{
  this.grdPosts.DataContext = ((Blog)cboBlogs.SelectedItem);
}
```

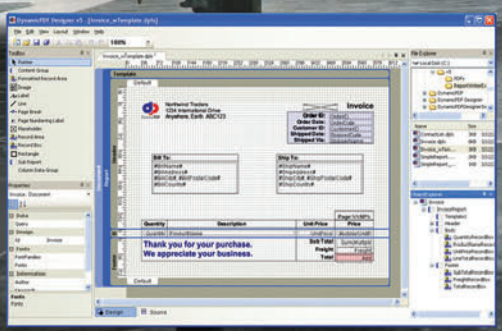
Figure 11 **Query Interceptor**

```
// returns only public posts and posts owned by the current user
[QueryInterceptor("Posts")]
public Expression<Func<Post, bool>>OnPostQuery()
{
  return p =>p.Public == true ||
  p.Blog.Owner.Equals(HttpContext.Current.User.Identity.Name);
}
// returns only the blogs the currently logged in user owns
[QueryInterceptor("Blogs")]
public Expression<Func<Blog, bool>>OnBlogQuery()
{
  return b =>
  b.Owner.Equals(HttpContext.Current.User.Identity.Name);
}
```

Speed • Reliability • Precision • Agility

DynamicPDF…Proven .NET Components for Real-Time PDFs

DynamicPDF



Layout reports in DynamicPDF Designer with its Visual Studio look and feel.

• Easy-to-use • Highly efficient
• Industry leading support • Huge feature set

Try it FREE today!

Experience our fully functional, never expiring evaluation and community editions.

DynamicPDF Suite v6.0 for .NET

Our easy-to-use tools integrate with ASP.NET and ADO.NET allowing for the quick, real-time generation of PDF documents and reports. For easy maintenance and deployment, our most popular tools are now available as a bundled suite.

DynamicPDF Generator v6.0 for .NET

• Linearize/Fast Web View • JavaScript • Encryption/Security • PDF/X-1a • PDF/A-1a/b • Create Tagged PDFs • Interactive Form Fields • Over 50 Ready-To-Use Page Elements Including 22 Bar Codes and Charting • Digital Signatures

DynamicPDF Merger v6.0 for .NET

• Merge • Stamp • Append • Split • Password/Security • Form-Fill • Outline and Annotation Preservation • Place, Rotate, Scale and Clip Pages • Decryption

DynamicPDF ReportWriter v6.0 for .NET

• GUI Report Designer • Event Driven • Recursive Sub-Reports • Use PDF Templates • Automatic Pagination • Placeholders • Record Splitting and Expansion • Column Support • Full DynamicPDF Merger and Generator Integration

Also, check out our newest product, DynamicPDF PrintManager for .NET.

ceTe Software has been delivering quality software applications and components to our customers for over 10 years. Our DynamicPDF product line has proven our commitment to delivering innovative software components and our ability to respond to the changing needs of software developers. We back our products with a first class support team trained to provide timely, accurate and thorough responses to any support needs.

ceTesoftware
INFINITE POSSIBILITIES

www.cete.com
info@cete.com

800.631.5006
+1 410.772.8620

This method will be called every time the item currently selected in the ComboBox is changed.

The last item to be hooked up on the Silverlight application is the Save Button, which is enabled by adding a btnSave_Click method that calls SaveChanges on the DataServiceContext, as seen in **Figure 14**.

Server-Driven Paging

Often there's a need to limit the total number of results a server will return for a given query, to avoid having an application accidentally pull back an extremely large amount of data. Server-driven paging in WCF Data Services 4.0 allows a service author to set per-collection limits on the total number of entities a service returns for each request by setting the SetEntitySetPageSize property in the InitializeService method for each collection of entities. In addition to limiting the number of entities returned for each request, the data service provides the client with a "next link"—a URI specifying how the client is to retrieve the next set of entities in the collection, in the form of the AtomPub<link rel="next"> element.

Going back to the MyBlog example, I'll set the SetEntitySetPageSize property on my service for the Posts EntitySet to five results:

```
config.SetEntitySetPageSize("Posts", 5);
```

This will limit the number of entities returned when the service is queried for Posts. I'm setting the SetEntitySetPageSize property to a small number here to illustrate how the feature works; generally an application would set a limit that most clients would not run into (rather the client would use $top and $skip to control the amount of data requested at any time).

I'll also add a new button to the application to allow the user to request the next page of Posts from the service. This snippet shows the btnMorePosts_Click method accessing the next set of Posts:

```
private void btnMorePosts_Click(object sender, RoutedEventArgs e)
{
  Blog curBlog = cboBlogs.SelectedItem as Blog;
  curBlog.Posts.LoadCompleted += new
    EventHandler<LoadCompletedEventArgs>(Posts_LoadCompleted);
  curBlog.Posts.LoadNextPartialSetAsync();
}
```

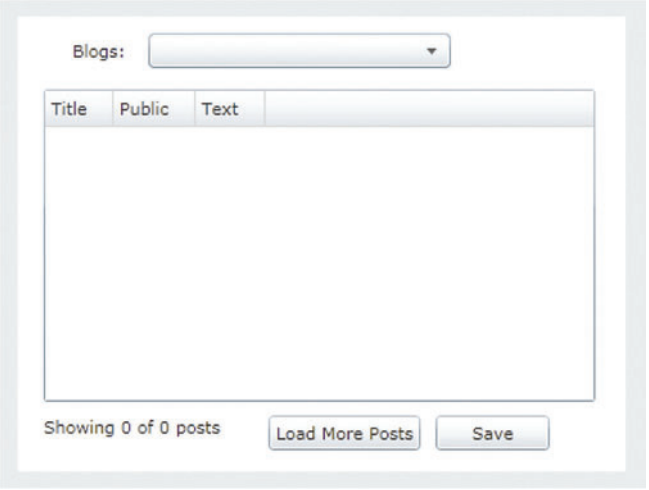


Figure 12 **Basic Layout of MyBlog Silverlight Administrator Application**

Row Count

One of the most requested features after the release of ADO.NET Data Services in Visual Studio 2008 SP1 was the ability to determine the total number of entities in a set without the need to pull them all back from the database. In WCF Data Services 4.0, we added the Row Count feature to do this.

When creating a query on the client, the IncludeTotalCount method can be called to include the count tag in the response. The value can then be accessed, as shown in **Figure 15**, using the TotalCount property on the QueryOperationResponse object.

Projections

Another of the most requested features in WCF Data Services 4.0 is Projections, the ability to specify a subset of an entity's properties to be returned from a query, allowing applications to optimize for band-

Figure 13 **MainPage.xaml and MainPage.xaml.cs**

MainPage.xaml

```
<Grid x:Name="LayoutRoot" Background="White" Width="618">
  <data:DataGrid Name="grdPosts" AutoGenerateColumns="False"
  Height="206" HorizontalAlignment="Left"Margin="17,48,0,0"
  VerticalAlignment="Top" Width="363" ItemsSource="{Binding Posts}">
    <data:DataGrid.Columns>
      <data:DataGridTextColumn Header="Title" Binding="{Binding Title}"/>
      <data:DataGridCheckBoxColumn Header="Public"
      Binding="{Binding Public}"/>
      <data:DataGridTextColumn Header="Text"
      Binding="{Binding PostContent}"/>
    </data:DataGrid.Columns>
  </data:DataGrid>
  <Button Content="Save" Height="23" HorizontalAlignment="Left"
  Margin="275,263,0,0" Name="btnSave" VerticalAlignment="Top"
  Width="75" Click="btnSave_Click_1" />
  <ComboBox Height="23" HorizontalAlignment="Left"
  Margin="86,11,0,0" Name="cboBlogs" VerticalAlignment="Top"
  Width="199" ItemsSource="{Binding}" DisplayMemberPath="Name"
  SelectionChanged="cboBlogs_SelectionChanged" />
  <dataInput:Label Height="50" HorizontalAlignment="Left"
  Margin="36,15,0,0" Name="label1"
  VerticalAlignment="Top"Width="100" Content="Blogs:" />
  <dataInput:Label Height="17" HorizontalAlignment="Left"
  Margin="17,263,0,0" Name="lblCount" VerticalAlignment="Top"
  Width="200" Content="Showing 0 of 0 posts"/>
  <Button Content="Load More Posts" Height="23"
  HorizontalAlignment="Left" Margin="165,263,0,0" Name="btnMorePosts"
  VerticalAlignment="Top" Width="100" Click="btnMorePosts_Click" />
</Grid>
```

MainPage.xaml.cs

```
public MainPage()
{
  InitializeComponent();
  svc = new BlogContext(new Uri("/BlogService.svc", UriKind.Relative));
  blogs = new DataServiceCollection<Blog>(svc);
  this.LayoutRoot.DataContext = blogs;
  blogs.LoadCompleted +=
  new EventHandler<LoadCompletedEventArgs>(blogs_LoadCompleted);
  var q = svc.Blogs.Expand("Posts");
  blogs.LoadAsync(q);
}
void blogs_LoadCompleted(object sender, LoadCompletedEventArgs e)
{
  if (e.Error == null)
  {
    if (blogs.Count> 0)
    {
      cboBlogs.SelectedIndex = 0;
    }
  }
}
```

Figure 14 **Saving Changes Back to the Database**

```
private void btnSave_Click_1(object sender, RoutedEventArgs e)
{
  svc.BeginSaveChanges(SaveChangesOptions.Batch, OnChangesSaved, svc);
}
private void OnChangesSaved(IAsyncResult result)
{
  var q = result.AsyncState as BlogContext;
  try
  {
    // Complete the save changes operation
    q.EndSaveChanges(result);
  }
  catch (Exception ex)
  {
    // Display the error from the response.
    MessageBox.Show(ex.Message);
  }
}
```

Figure 15 **Using Row Count**

```
private void cboBlogs_SelectionChanged(object sender,
SelectionChangedEventArgs e)
{
  Blog curBlog = this.cboBlogs.SelectedItem as Blog;
  this.grdPosts.DataContext = curBlog;
  var q = (from p in svc.Posts.IncludeTotalCount()
  where p.BlogBlogID == curBlog.ID
  select p) as DataServiceQuery<Post>;
  curBlog.Posts.LoadCompleted += new
  EventHandler<LoadCompletedEventArgs>(Posts_LoadCompleted);
  curBlog.Posts.LoadAsync(q);
}
void Posts_LoadCompleted(object sender, LoadCompletedEventArgs e)
{
  if (e.Error == null)
  {
    Blog curBlog = cboBlogs.SelectedItem as Blog;
    totalPostCount = e.QueryOperationResponse.TotalCount;
    string postsCount = string.Format("Displaying {0} of {1} posts",
    curBlog.Posts.Count, totalPostCount);
    this.lblCount.Content = postsCount;
    curBlog.Posts.LoadCompleted -= Posts_LoadCompleted;
  }
}
```

width consumption and memory footprint. In Visual Studio 2010, the Data Services URI format has been extended to include the $select query option, enabling clients to specify the subset of properties to be returned by the query. For example, with MyBlog, I could query for all Posts and project only Title and PostContent using the following URI: BlogService.svc/Posts?$select=Title,PostContent. On the client side, you can also now use LINQ to query with projections.

Learning More

This article covers just enough to get you started with Entity Framework 4.0 and WCF Data Services 4.0 in Visual Studio 2010. There are a number of other topics and new features you may be interested in. For more information, check out the MSDN Data Development center at msdn.microsoft.com/data. ■

Elisa Flasko *is a program manager in the Data Programmability team at Microsoft, working on the ADO.NET Entity Framework, WCF Data Services, M, Quadrant and SQL Server Modeling Services technologies. She can be reached at blogs.msdn.com/elisaj.*

Thanks to the following technical experts for reviewing this article: *Jeff Derstadt and Mike Flasko*

CHARLES PETZOLD

Projection Transforms Sans Math

In pretty much any graphics system, transforms constitute the most important feature that doesn't actually draw anything. Instead, transforms alter the appearance of visual objects by modifying coordinates with mathematical formulas generally expressed as a matrix multiplication.

Figure 1 **Text with a Taper Transform**

The RenderTransform property defined by the UIElement has been in Silverlight from its beginning, and before that, in the Windows Presentation Foundation (WPF). Because the property is defined by UIElement, you can use it with graphical objects as well as text, controls and media. Simply set the RenderTransform to an object of type TranslateTransform, ScaleTransform, RotateTransform, SkewTransform, MatrixTransform (for complete control over the transform matrix) or a TransformGroup for a combination of multiple transforms.

The types of transforms you set with RenderTransform are all examples of two-dimensional (2D) *affine* transforms. Affine transforms are very well behaved and just a little dull: Straight lines are always transformed to straight lines, ellipses are always transformed to ellipses and squares are always transformed to parallelograms. If two lines are parallel before the transform, they're still parallel after the transform.

Pseudo 3D

Silverlight 3 introduced a new UIElement property named Projection that allows setting *non-affine* transforms on graphical objects, text, controls and media. Non-affine transforms do not preserve parallelism.

The type of non-affine transform allowed in Silverlight 3 is still represented by a matrix multiplication, and it still has restrictions on what it can do. Straight lines are always transformed to straight lines, and a square is always transformed into a simple convex quadrilateral. By "quadrilateral," I mean a four-sided figure (also called a tetragon or quadrangle); by "simple," I mean that the sides don't intersect except at their vertices; by "convex," I mean that the internal angles at each vertex are less than 180 degrees.

This type of non-affine transform is useful for creating taper transforms, where opposite sides of a square or rectangle taper somewhat in one direction. **Figure 1** shows some text with a taper transform realized through a very simple Projection property setting.

The text appears to be somewhat three dimensional (3D) because the tail end seems further away from our eyes—an effect called a perspective projection.

In a sense, the Projection property gives Silverlight a little bit of "pseudo 3D." It's not a real 3D system, because there's no way to define objects in 3D space, no concept of cameras, lights or shading and—perhaps most important—no clipping of objects based on their arrangement in 3D space.

Nevertheless, working with the Projection transform requires the programmer to begin thinking about three dimensions and especially about 3D rotation. Fortunately, the developers of Silverlight have made some common and simple use of the Projection property fairly easy.

The Easier Approach

You can set the Projection property to either a Matrix3DProjection object or a PlaneProjection object. The Matrix3DProjection property defines only one object, but it's a 4x4 Matrix3D structure, which requires lots of mathematics. (For some approaches to this structure, see the blog entries on my Web site—charlespetzold.com—dated July 23, 2009, and July 31, 2009.)

But in this article, I've promised myself to avoid mathematics for the most part, which means I'll be sticking with the PlaneProjection class. Although the class defines 12 settable properties, I'll be focusing on only six of them.

In the downloadable source code for this article is PlaneProjectionExperimenter, which lets you interactively experiment with these six properties. **Figure 2** shows the program in action. You can run it by compiling the downloadable program, or you can run it at my Web site at charlespetzold.com/silverlight/PlaneProjectionDemos, along with all the other programs in this article. For now, ignore the blue dot in the middle.

The three crucial properties of PlaneProjection are RotationX, RotationY and RotationZ, which you can change using the three ScrollBars. These properties assume a 3D coordinate system where X values increase to the right and Y values increase going down the screen. (This is consistent with the normal Silverlight coordinate system but not with typical 3D systems, where values of Y usually increase going up.) Increasing values of Z seem to come out of the screen toward the viewer. These three properties cause rotation around the three axes.

Code download available at code.msdn.microsoft.com/mag201004UIF.

Does your Team do more than just track bugs?

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

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

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

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

Native Smart Card Login Support including Government and DOD

New in Team 2.11

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

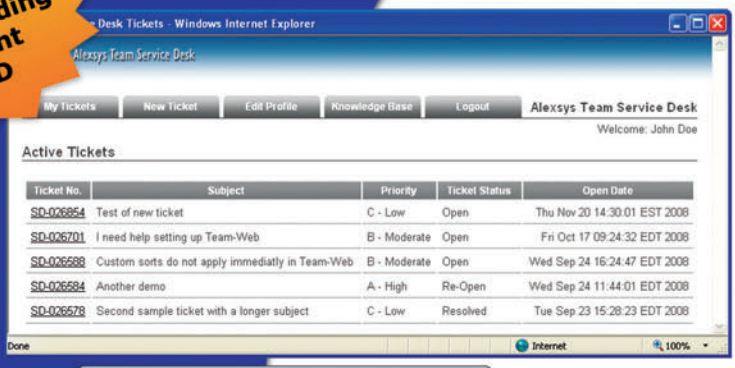
Service Desk Features

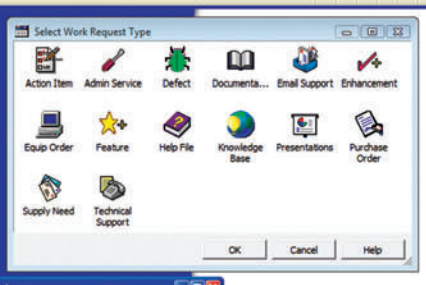
- Fully Secure
- Unlimited Users Self Registered or Active Directory
- Integrated into Your Web Site
- Fast/AJAX Dynamic Content
- Unlimited Service Desks
- Visual Service Desk Builder

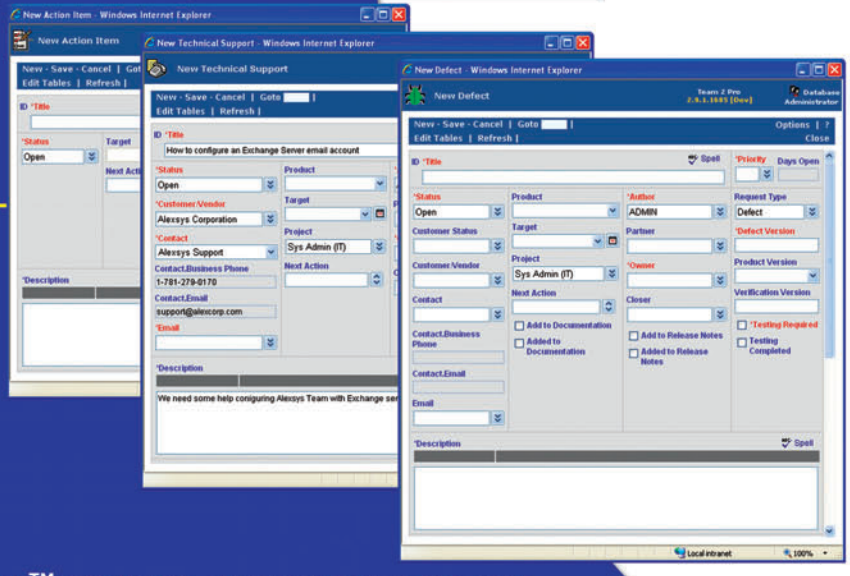
Team 2 Features

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

ZDNet Editors' Pick







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

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

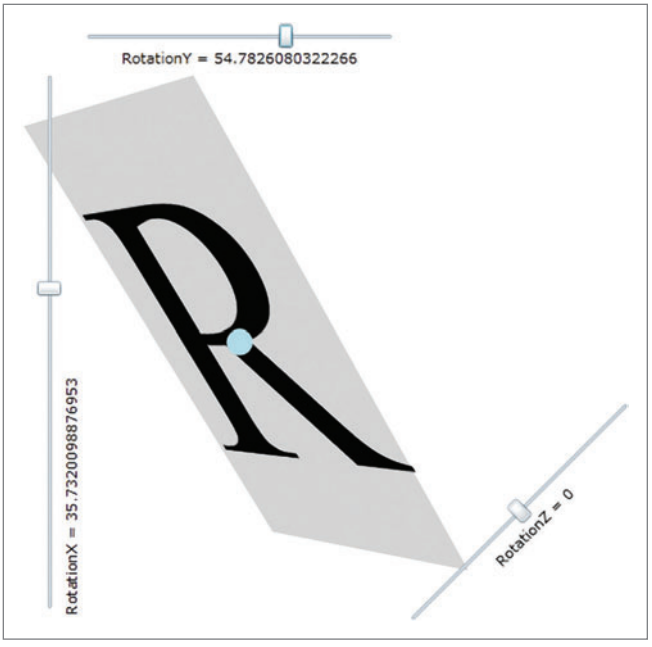


Figure 2 **The PlaneProjectionExperimenter Program**

The ScrollBars for X and Y are positioned perpendicularly to the axis of rotation. For example, the vertical ScrollBar on the left rotates the figure around the X axis, which runs horizontally through the center of the letter. The top and bottom of the letter seem to flip toward or away from the viewer.

At first, it's best to try each axis of rotation independently of the others and refresh your browser between experimentations. You can anticipate the direction of rotation using the right-hand rule. Point your thumb in the direction of the positive axis. (For X, that's to the right; for Y it's down; for Z, it's toward you.) The curve that your other fingers make indicates the direction of rotation for positive rotation angles. Negative angles rotate in the opposite direction.

A composite rotation depends on the order in which the individual rotations are applied. Using PlaneProjection sacrifices some flexibility in these rotations. PlaneProjection always applies RotationX first, then RotationY and finally RotationZ. How can we tell? Try leaving RotationZ at 0 and manipulate RotationX and RotationY. You'll see that RotationX always rotates the letter around the horizontal axis of the letter itself, whereas RotationY rotates the letter around the vertical axis of the *window*, meaning that RotationY is applied to the letter already rotated by the RotationX angle. Now, with RotationX and RotationY set to anything, manipulate RotationZ. This rotation is also relative to the window, and doesn't change the appearance of the letter at all.

In real life, you can simply set one property of Projection and get a reasonable result. The text in **Figure 1** is part of the TaperText project and was displayed using the following XAML:

```
<TextBlock Text="TAPER"
           FontFamily="Arial Black"
           FontSize="144"
           HorizontalAlignment="Center"
           VerticalAlignment="Center">
    <TextBlock.Projection>
        <PlaneProjection RotationY="-60" />
    </TextBlock.Projection>
</TextBlock>
```

As with RenderTransform, Projection doesn't affect layout. The layout system sees an un-transformed and un-projected element.

Of course, the RotationX, RotationY and RotationZ properties are all backed by dependency properties, so they can also become animation targets.

The FlipPanel

With just this much knowledge of PlaneProjection, it's possible to code a "flip panel," which is a technique to minimize an application

Figure 3 **The FlipPanel.xaml.cs File**

```
using System;
using System.Windows;
using System.Windows.Controls;

namespace FlipPanelDemo
{
    public partial class FlipPanel : UserControl
    {
        public static readonly DependencyProperty Child1Property =
            DependencyProperty.Register("Child1",
                typeof(UIElement),
                typeof(FlipPanel),
                new PropertyMetadata(null, OnChild1Changed));

        public static readonly DependencyProperty Child2Property =
            DependencyProperty.Register("Child2",
                typeof(UIElement),
                typeof(FlipPanel),
                new PropertyMetadata(null, OnChild2Changed));

        public FlipPanel()
        {
            InitializeComponent();
        }

        public UIElement Child1
        {
            set { SetValue(Child1Property, value); }
            get { return (UIElement)GetValue(Child1Property); }
        }

        public UIElement Child2
        {
            set { SetValue(Child2Property, value); }
            get { return (UIElement)GetValue(Child2Property); }
        }

        public void Flip()
        {
            flipStoryboard.Begin();
        }

        public void FlipBack()
        {
            flipBackStoryboard.Begin();
        }

        static void OnChild1Changed(DependencyObject obj,
                    DependencyPropertyChangedEventArgs args)
        {
            (obj as FlipPanel).child1Container.Content = args.NewValue;
        }

        static void OnChild2Changed(DependencyObject obj,
                    DependencyPropertyChangedEventArgs args)
        {
            (obj as FlipPanel).child2Container.Content = args.NewValue;
        }
    }
}
```

Figure 4 **The FlipPanel.xaml File**

```
<UserControl x:Class="FlipPanelDemo.FlipPanel"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/
presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
    <UserControl.Resources>
        <Storyboard x:Name="flipStoryboard">
            <DoubleAnimation Storyboard.TargetName="planeProjection1"
                             Storyboard.TargetProperty="RotationX"
                             To="90"
                             Duration="0:0:0.5" />

            <DoubleAnimation Storyboard.TargetName="planeProjection2"
                             Storyboard.TargetProperty="RotationX"
                             To="0"
                             BeginTime="0:0:0.5"
                             Duration="0:0:0.5" />
        </Storyboard>

        <Storyboard x:Name="flipBackStoryboard">
            <DoubleAnimation Storyboard.TargetName="planeProjection1"
                             Storyboard.TargetProperty="RotationX"
                             To="0"
                             BeginTime="0:0:0.5"
                             Duration="0:0:0.5" />
            <DoubleAnimation Storyboard.TargetName="planeProjection2"
                             Storyboard.TargetProperty="RotationX"
                             To="-90"
                             Duration="0:0:0.5" />
        </Storyboard>

    </UserControl.Resources>

    <Grid x:Name="LayoutRoot" Background="White">
        <ContentControl Name="child1Container">
            <ContentControl.Projection>
                <PlaneProjection x:Name="planeProjection1" />
            </ContentControl.Projection>
        </ContentControl>
        <ContentControl Name="child2Container">
            <ContentControl.Projection>
                <PlaneProjection x:Name="planeProjection2"
                                 RotationX="-90" />
            </ContentControl.Projection>
        </ContentControl>
    </Grid>
</UserControl>
```

footprint onscreen by organizing controls on the front and back of a panel (or so it seems). In WPF, a FlipPanel control requires switching back and forth between 2D and 3D. In Silverlight, the FlipPanel becomes quite simple.

The FlipPanelDemo project includes a FlipPanel control derived from UserControl. The code part shown in **Figure 3** defines two new properties named Child1 and Child2 of type UIElement, and public methods named Flip and FlipBack.

The XAML part in **Figure 4** shows how Child1 and Child2 occupy the same space and have Projection transforms applied. In the initial position, the PlaneProjection transform for Child2 has RotationX set to -90 degrees, which means that it's at right angles to the viewer and is effectively invisible. The "flip" animations simply swing Child1 out of view by animating RotationX to 90; they swing Child2 into view by animating RotationX to 0. The "flip back" animations reverse those actions.

Commonly, Child1 and Child2 would be set to panels of some sort covered with controls. In the FlipPanelDemo program, Child1 and Child2 simply have different colors and a single button to trigger the flip. The flipping action seems to be more comforting to the user than navigating to a new page because it implies that a set of controls aren't disappearing irrevocably but can easily be retrieved.

Center of Rotation

All rotation is relative to a center. Rotation in two dimensions is relative to a point. Rotation in three dimensions is relative to a line in 3D space—often referred to as an "axis of rotation." For convenience—and perhaps to avoid introducing 3D lines and 3D vectors into Silverlight—the projection transform is considered to be relative to a 3D point.

The PlaneProjection class supports changing the center of rotation using three properties:

- CenterOfRotationX (relative coordinate; default is 0.5)
- CenterOfRotationY (relative coordinate; default is 0.5)
- CenterOfRotationZ (absolute coordinate; default is 0)

Let's look at the first two first. Like the RenderTransformOrigin property, these values are relative to the upper-left corner of the element being transformed. They differ in that the default value of RenderTransformOrigin is the point (0, 0), whereas the default values for PlaneProjection cause the rotations to be centered on the element.

In PlanProjectionExperimenter, you can change CenterOfRotationX and CenterOfRotationY using the blue dot. (A tooltip indicates the current values.) You'll probably notice right away that CenterOfRotationX does not affect rotation around the X axis, and CenterOfRotationY does not affect rotation around the Y axis.

If you make CenterOfRotationX equal to 0, rotation around the Y axis is relative to the left side of the element; similarly, if you set it to 1, rotation is around the right side. You can make the value less than 0 or greater than 1 for some very wide swings. The element seems to get very close to the viewer and go very far away.

Try this: Make CenterOfRotationY approximately equal to 0. Now, set RotationX equal to 90 degrees or thereabouts. You'll notice that the element is still visible. In the FlipPanel, a rotation of 90 degrees or -90 degrees causes the element to be invisible because it's viewed on its edge. The mathematical calculations going on inside the PlaneProjection class assume that the viewer's eye (or the metaphorical camera) is always aligned in the center of the element looking straight back in the negative-Z direction. If something is off-center and rotated back 90 degrees, it's still going to be visible from the center of the element.

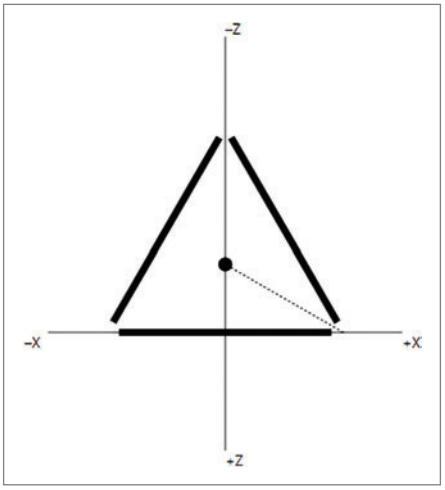


Figure 5 **A Proposed Three-Panel Carousel**

The CenterOfRotationZ axis is handled differently from X and Y. It's in Silverlight absolute coordinates rather than relative coordinates, and the default is 0. In PlaneProjectionExperimenter, you can change this value using the mouse wheel. The blue dot grows and shrinks to represent the change in the property.

Rotation with a non-default CenterOfRotationZ is probably the hardest to visualize mentally, so it's worth the time to experiment a bit.

The element being projected is assumed to sit in the XY plane—that is, the plane in 3D space where Z equals 0. If you leave the CenterOfRotationZ property at its default value of 0 and manipulate RotationX or RotationY, parts of the letter get larger as they move into positive-Z space, and parts get smaller as they move into negative-Z space. Now increase CenterOfRotationZ to a value of about 200 and manipulate RotationY. The letter will get larger because it's rotating around the center where Z equals 200 from an area of space where Z equals 0 to the area where Z equals 400. When you set CenterOfRotationZ to 500 or greater, the whole thing stops working well because the internal mathematics of PlaneProjection assume that the viewer (or camera) is located 1,000 units from XY plane. With a rotation center of 500, the element is actually being projected behind the camera.

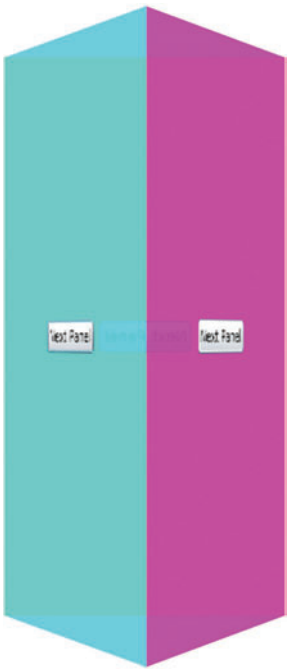


Figure 6 **The ThreePanelCarousel Program as it's Spinning**

Three-Panel Carousel

Can we make something similar to a FlipPanel with three sides rather than just two? Yes, but it will require a tiny bit of trigonometry and a whole lot of messing around with Z indices.

Figure 5 shows a top view of what I envision. It's a view from a negative position on the Y axis—above the monitor, so to speak—looking down. The thick lines represent the three panels. All these panels are actually positioned in the same spot. Only the Projection transform causes them to appear in different locations. The visible one in front has no rotations applied to it. The other two have been subjected to rotations where RotationY is set to 120 (for the one on the right) and -120 degrees (on the left). Both rotations are centered on a negative CenterOfRotationZ value, which corresponds to the black dot. What is that value?

If you connect that dot with a dotted line to the vertex at the right, the dotted line makes an angle of 30 degrees with the X axis. The tangent of 30 degrees is 0.577. That's the ratio of the distance of the dot to the X axis, divided by half the width of the panel. If the panel is 200 units wide, that dot is at -57.7. Set CenterOfRotationZ to that value.

Now, to rotate the carousel from one panel to the next, just animate RotationY to increase it by 120 degrees for all three panels.

Well, not exactly. Even though we're picturing these panels in 3D space, they really still exist in 2D space, and they're actually stacked on top of one another. As they're rotating, the Canvas.ZIndex attached property of each of the three panels must be changed to reorder the panels. This is how elements seemingly can be moved "in front of" or "behind" other elements.

Let's assume rotation is clockwise based on the **Figure 5** diagram. To begin, the panel in front should have a Z index of 2 (foreground), the one on the right has a Z index of 1 and the one on the left has a Z index of 0 (background). Now start the 120-degree rotation. Halfway through—that is, when each panel has been rotated 60 degrees and two panels are equally visible—the Z indices must be changed. The panel moving into view should have its Z index set to 2, and the one moving out of view should have its Z index set to 1.

This is all implemented in the ThreePanelCarousel project. In actual use, of course, the three panels would be covered with controls; in this demonstration program, they just have different colors and one button. I've left the panels partially transparent so you can see what's going on. **Figure 6** shows the panels in action.

You may wonder if it's possible to apply a single projection rotation to the composite group of three panels. No, it's not. Multiple projection transforms can't be compounded over child elements.

Other Effects

As you experiment with the Projection property, don't forget about other effects to "enhance" the 3D-ishness of your application. A little shadow always helps, and it's easy to simulate a shadow—even a moving shadow—with a gradient brush.

The AsTheDaysGoBy program uses a combination of animations to simulate the days flying off a daily desk calendar, as if in an old movie to suggest the passing of time. The calendar page has its Projection property set to a PlaneProjection object with the CenterOfRotationX and CenterOfRotationY properties both set to 0. RotationX and RotationY are both animated to move the page up and to the left while an animated gradient brush sweeps up the page. Finally, the page just seems to disappear as its Opacity property is animated to 0. **Figure 7** shows the program in action.

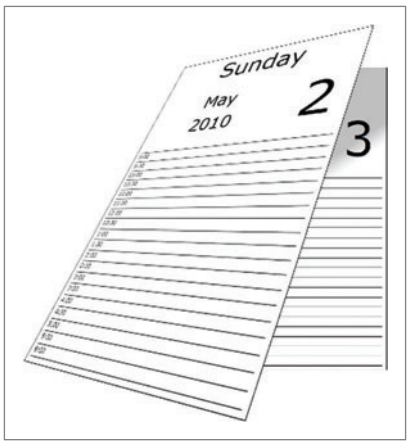


Figure 7 **The AsTheDaysGoBy Program**

The animation lasts a second, so the program goes through 60 days every minute, or about 10 years per hour. After about a month, the program will be approaching the maximum DateTime value, and it will soon terminate with an exception. Until then, enjoy. ■

Charles Petzold *is a long-time contributing editor to* MSDN Magazine. *His most recent book is "The Annotated Turing: A Guided Tour Through Alan Turing's Historic Paper on Computability and the Turing Machine" (Wiley, 2008). Petzold blogs on his Web site charlespetzold.com.*

DAVID S. PLATT

In Praise of Dumbing Down

A typical geek hurried up to me after a talk I'd given, elbowing other attendees aside to get first crack at me. Red-faced, he spluttered, "You just spent an hour telling us to dumb down our programs. That's awful, and you ought to be shot."

I had praised a particular automated disk backup program. Like seat belts or birth control, disk backup programs only work if you use them. The hassle of configuring and running backup (or birth control) is the main reason it fails. The program I was praising avoids that through seamless automation. The user doesn't have to touch it after installation, ever. He doesn't have to specify which files to back up; the program automatically copies all data files to a remote server, then automatically copies every new or changed file thereafter. Users don't control very much: they don't supply their own encryption keys, don't set compression levels, don't even schedule when the program runs; it's always there in the background. The geek objected strongly to this lack of control, not just saying he wouldn't buy it, but insisting that it was sinful and shouldn't exist and I was evil for praising it.

Is this smart or dumb?

I have more experience being heckled than most hecklers have doing it, so I slammed right back at him: "Was it dumbing down when Ford removed the spark timing lever from the Model T? Was it dumbing down when they replaced the hand crank with a self-starter?"

Geek: "No, those are reasonable. Anyway, cranking the engine is a hardware problem."

Me: "Does your car have GPS?"

Geek: "Yeah, and it's really cool!"

Me: "Is it dumbing down when it gives you directions—'Move left one lane and take the middle fork'? No, you say, 'Wow, one-meter precision, cool.' How about when the power seat remembers your reclining settings, or the stereo remembers the songs you like?"

The geek didn't give up: "But it takes brains to drive a car, and to use a computer, and if you're not smart enough, you shouldn't be doing it."

Perhaps, but less every year for both, and I welcome both decreases. I was taught years ago to pump the brakes when my car skidded. Today's cheap automatic anti-lock brakes do that far better than the best human drivers could. And today's automated traction control systems greatly reduce skidding. Does that make the world a better place or a worse one? I say better, unless you own a body shop.

So don't get me started on the term "dumbing down." It's a loaded, judgmental term that reveals our prejudices. We geeks are proud of ourselves for being smart. We respect people who can remember command keys without a keyboard template, or shout out each line of the boot script half a second before it appears on the screen. The most toxic word in our vocabulary is "stupid."

Some geeks today resent making their programs easy to use, wanting users to struggle as we had to. It's like your Depression-era grandparents resenting today's coddled youth: "When I was a boy, we didn't have air-conditioned school buses with cushy reclining seats and personal video screens and 87 satellite dish channels. No! We had to walk 10 miles through the snow, uphill both ways …" There's no place for that attitude today in a profitable company. If your car could magically drive itself, or we had transporters so we didn't need cars, wouldn't you use them?

> The hassle of configuring and running backup (or birth control) is the main reason it fails.

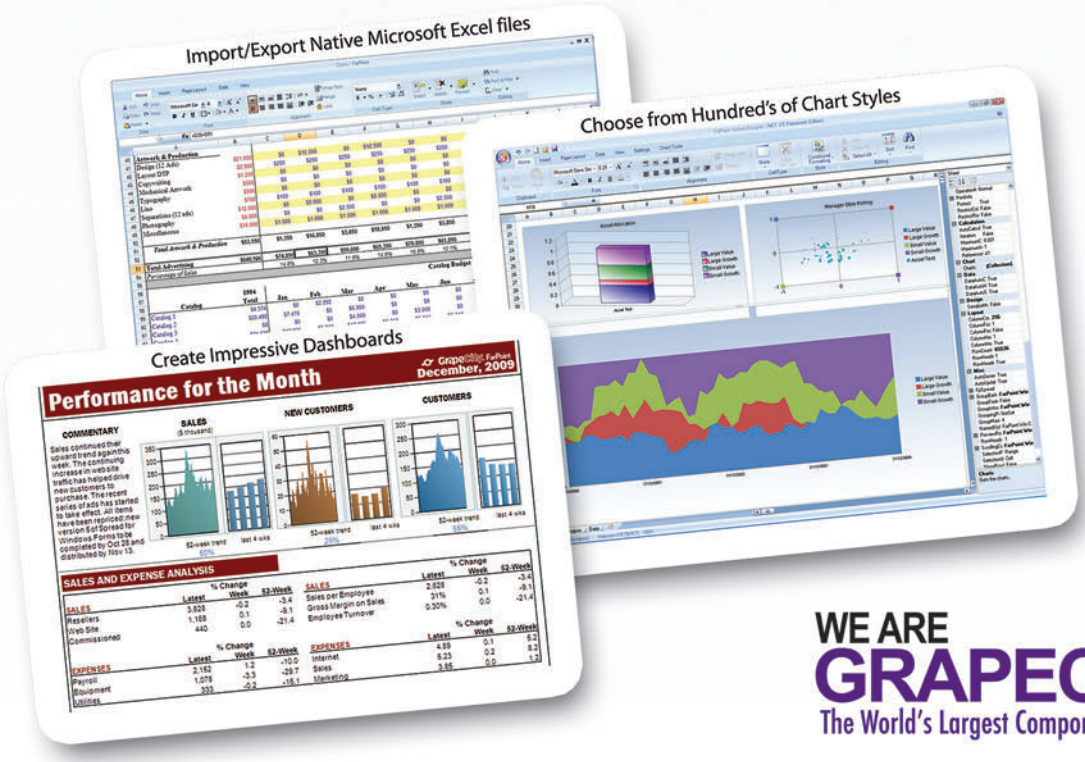
Of course, dumbing down has to be done well, which requires a smart interaction designer—someone who knows how to think like a user, to make main use cases easy by omitting obscure edge cases, to choose helpful default values, to explain choices in terms of the user's thoughts rather than the program's implementation. The designers of the Windows Move Maker did this well, as I wrote in my blog posting at suckbusters2.blogspot.com/2007/02/another-application-that-just-works-at.html.

The "dumbed down" backup program that so infuriated my listener means that many more users actually will back up their data, so they won't lose it when their disks crash or their offices burn down or their laptops get stolen. I find that extremely smart. ■

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" and "Introducing Microsoft .NET." 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.*

GrapeCity

WE ARE SPREADSHEETS



WE ARE GRAPECITY

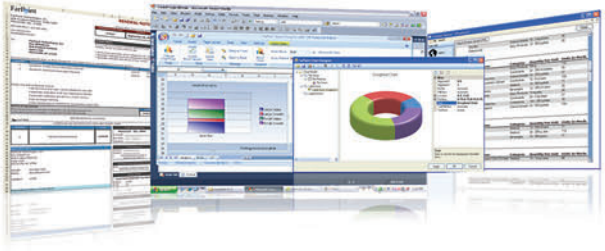
The World's Largest Component Vendor

Integrate the World's #1 Selling Microsoft® Excel® compatible Spreadsheet Components into your .NET Applications.

- ✓ Embed the familiar spreadsheet user experience without requiring Excel
- ✓ Provide advanced analysis for your Business Intelligence needs
- ✓ Visualize your data using the new Chart Styles
- ✓ Rapid development using the enhanced Spread Designer and Quick Start Wizards
- ✓ Used by over 100,000 developers

PowerTools	SPREAD 5	ACTIVEREPORTS 6	ANALYSIS 1
Deep Functionality	100%	100%	100%
Award Winning	✓	✓	New
Category Leader	✓	✓	✓
Industry's Best Support	✓	✓	✓
Proven Track Record	✓	✓	✓

SPREAD 5
Windows Forms



www.FarPointSpread.com & www.DataDynamics.com

Download your free evaluation today at www.FarPointSpread.com

Spreadsheets Reporting Analysis