THE MICROSOFT JOURNAL FOR DEVELOPERS

SEPTEMBER 2014 VOL 29 NO 9

msdn magazine

THE MICROSOFT JOURNAL FOR DEVELOPERS

SEPTEMBER 2014 VOL 29 NO 9

msdn magazine

Microsoft Azure Mobile Services.....18

COLUMNS

Microsoft

msdn magazine

SEPTEMBER 2014 VOLUME 29 NUMBER 9

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

MICHAEL DESMOND

Six Degrees of Live! 360

A few months back we welcomed Lafe Low to the magazine as a features editor. Lafe has been a fixture in the dev conference group at 1105 Media, helping shepherd the family of Visual Studio Live! and Live! 360 developer events. So when erstwhile Technical Editor David Ramel shuffled off to a new position with the Web editorial team at 1105 Media, Lafe was a natural choice to step into the gap.

Anyone who has met Lafe knows that the man is wired for 220. He brings amazing energy, humor and optimism to everything he does, and that perspective shines through in his work. So it should be little surprise to learn that Lafe took to his new role here at the magazine the same way he attacks the chutes at Tuckerman Ravine in New Hampshire—with an almost maniacal abandon.

> "Billy Hollis actually said it best: 'Industry newbies, they go to conferences for the presentations. Veterans, they go for the people and the parties.'"
>
> — *Ted Neward*

Lafe's arrival made me realize that *MSDN Magazine* has ties with the Live! 360 and Visual Studio Live! conferences that go back years. Senior Contributing Editor James McCaffrey was a frequent presenter at Visual Studio Live! events until about two years ago. Two other *MSDN Magazine* columnists—Ted Neward and Rachel Appel—are Live! 360 regulars today.

I asked McCaffrey about his experience at Visual Studio Live! and he said he valued presenting there because the events helped him recharge his (and I quote) "developer psychic energy."

Says McCaffrey: "I speak at, and enjoy, official Microsoft conferences like Build and TechEd, but many talks at those events are forward-looking and 'vision' messages—as they should be. The talks at Visual Studio Live! give me practical information I can put to use immediately."

Ted Neward has been writing The Working Programmer column in *MSDN Magazine* for more than four years now. He was presenting at Visual Studio Live! way back in 2004 (when it was owned by Fawcette Technical Publications), and he's now a regular on the Live! 360 circuit. While Neward praises the technical content of the events, he says the real value is the opportunity for attendees to forge lasting professional relationships.

"I contend that any attendee that goes to a conference and doesn't have a stack of business cards is an idiot," Neward says, tongue planted only half in cheek. "Billy Hollis actually said it best: 'Industry newbies, they go to conferences for the presentations. Veterans, they go for the people and the parties.'"

The next party takes place in Orlando, Nov. 17-21. Maybe you should meet (and party with) Ted Neward there.

Azure Insider Gets an Upgrade

The eagle-eyed among you might notice a change with our regular Azure Insider column. The long-standing column has moved into the feature well, to appear as an ongoing feature series under the expert management of Bruno Terkaly and Ricardo Villalobos. I'll let Ricardo take the mic, because he says it better than I:

"Considering the rapid and ongoing evolution of Microsoft Azure, it's critical to get the inside view from the engineers building the platform's infrastructure and services, as well as from those implementing solutions that address real-world scenarios. Based on the work and writing that we've done during the last two years for the Azure Insider column, cloud specialists worldwide will join our effort, collaborating with us in writing articles that will address common scenarios, offering practical solutions based on their experiences. Some of the areas where we will focus our efforts are high-scale database engines, global media distribution, scalable sensor-based analysis, machine learning and multi-platform mobile support."

Azure Insider is dead. Long live Azure Insider!

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

Data Quality Tools for Developers

A better way to build in data verification

Since 1985, Melissa Data has provided the tools developers need to enhance databases with clean, correct, and current contact data. Our powerful, yet affordable APIs and Cloud services provide maximum flexibility and ease of integration across industry-standard technologies, including .NET, Java, C, and C++. Build in a solid framework for data quality and protect your investments in data warehousing, business intelligence, and CRM.

- **Verify international addresses for over 240 countries**
- **Enhance contact data with phone numbers and geocodes**
- **Find, match, and eliminate duplicate records**
- **Sample source code for rapid application development**
- **Free trials with 120-day ROI guarantee**

Address Verification

Phone Verification

Email Verification

Geocoding

Matching/ Dedupe

Change of Address

Melissa Data.
Architecting data quality success.

MELISSA DATA®

www.MelissaData.com 1-800-MELISSA

DINO ESPOSITO

A Look at ClearScript

Years ago, I was fascinated by the prospect of hosting the entire VBScript engine of Active Server pages within a Visual Basic application. I could create an astonishing proof of concept for a company willing to sell editorial content on CD that would reuse existing Active Server pages content just outside local or remote Web servers.

It was the late 1990s. There was no Microsoft .NET Framework. There was no HTML5. Only a few of us were actively exploring the depths of Dynamic HTML, yet hosting the scripting engine couldn't have been easier. I had to reference an ActiveX control, publish my ActiveX objects within the scripting environment and I was ready to go.

More recently, a customer asked me about the most effective way to extract text files from SQL Server queries. That question was outside the usual range of topics I handle, so I was tempted to answer with something like, "Sorry, I don't know." However, I knew this customer was strong in database operations. I suspected there was more background behind the question.

The customer was regularly producing plain text files (mostly CSV and XML files) out of content stored in database tables within an instance of SQL Server. This annoyed its database staff, as requests were coming mostly from business with the usual urgency of business issues. There was no recurrent logic that could help create repeatable routines—at least in a SQL Server environment.

Ultimately, the customer was looking for a tool business people could program using easy script languages such as VBScript. The users needed controlled access to the databases for read-only purposes. Needless to say, the tool had to offer users a chance to easily create text files. This reminded me of the happy days of ActiveX and VBScript. I almost had a wave of regret when I heard about a relatively new library named ClearScript (clearscript.codeplex.com).

Integrate ClearScript into Windows Presentation Foundation

ClearScript lets you add scripting capabilities to a .NET application (as long as it uses the .NET Framework 4 or higher). ClearScript supports VBScript, JavaScript and V8. V8 is an open source JavaScript engine created by Google and integrated with Chrome. V8 has a high-performance JavaScript engine and fits well within multi-threading and asynchronous operation scenarios.

Code download available at msdn.microsoft.com/magazine/msdnmag0914.

The net effect of adding ClearScript to a .NET application is you can pass JavaScript or VBScript expressions to the engine and they'll be processed and run. Interestingly, you aren't limited to using plain script objects such as arrays, JSON objects and primitive types. You can integrate external JavaScript libraries and script-managed .NET objects.

Once you've integrated ClearScript into an application, all that remains is to let the library know about the objects it can script. This means you publish your own objects in the context of ClearScript and let authorized users load and run existing scripts or write new ones.

If you want to add a layer of customization and let the user add pieces of his own logic without incurring the costs of change requests, then ClearScript is necessary, but it might not be sufficient. ClearScript is just one piece of the jigsaw puzzle. You might want to provide a way for the user to manage his own scripts. Also, you should create some ad hoc objects that simplify common tasks like creating files.

Here's what I did to help a customer generate text and XML reports from a bunch of services exposed through a Web API back end. The main functional requirement was to let users create text files. For the proof of concept, I needed a shell application to host ClearScript. I opted for a Windows Presentation Foundation (WPF) application with a textbox to manually enter script code. Successive iterations added support for a default input folder and a UI to open/import existing script files. **Figure 1** shows the WPF sample application in action.

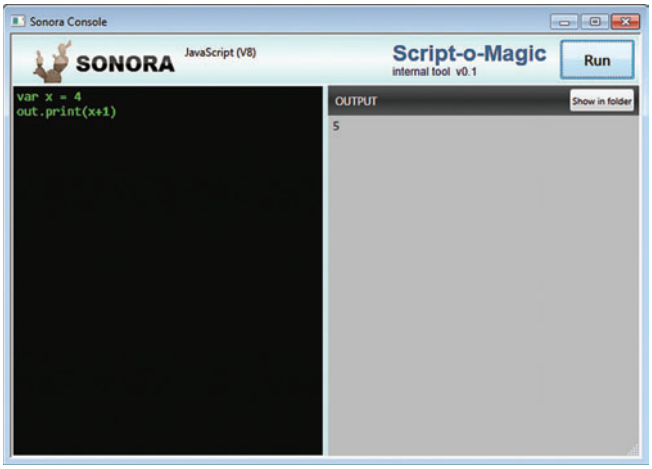


Figure 1 **A Sample Windows Presentation Foundation Application Hosting the ClearScript Engine**

Again, ClearScript is an open source project you can reference directly in your project by linking assemblies. You can also go through third-party NuGet packages, as shown in **Figure 2**.

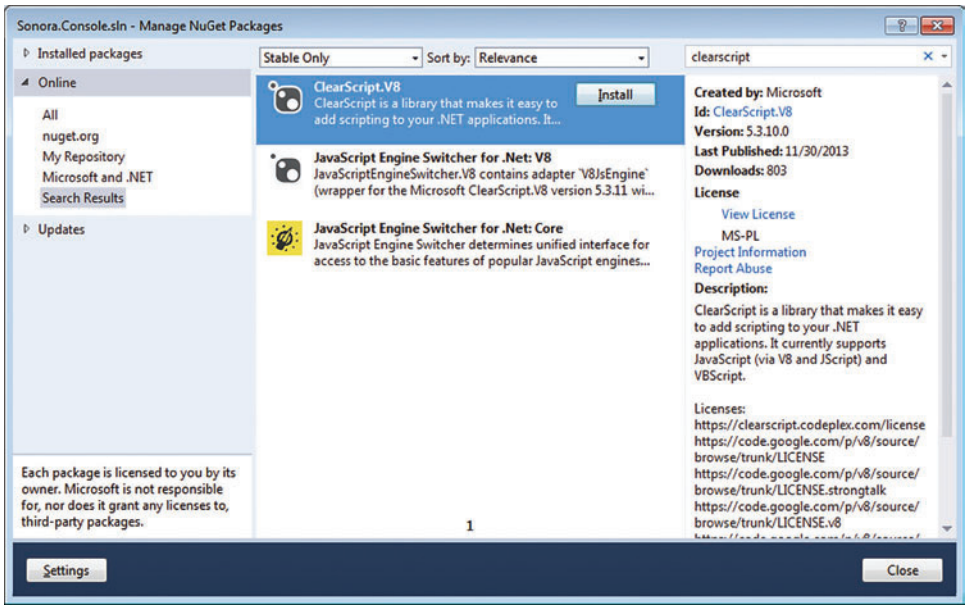


Figure 2 **You Can Install ClearScript via NuGet**

Initialize ClearScript

You'll have to do some work before you can use the scripting engine programmatically. But at the end of the day, once it's completely set up, the code in **Figure 3** is all you need to trigger script code execution.

The Confirm method belongs to the presenter class supporting the main view of the sample application. Trigger the method by any click on the Run button (visible in **Figure 1**). The SonoraConsole class you see referenced in the listing is just my own wrapper around the core classes of the ClearScript library.

Initializing the ClearScript engine takes place as the application starts up and is bound to the Startup event of the XAML Application class:

```
public partial class App : Application
{
  void Application_Startup(Object sender, StartupEventArgs e)
  {
    SonoraConsole.Initialize();
  }
}
```

The initialization can be as complex and sophisticated as you'd like, but it at least has to initialize the script engine of your selected language. You have to make the instance of the script engine available to the other parts of the application. Here's one possible approach:

```
public class SonoraConsole
{
   public static void Initialize()
   {
     ScriptEngine = new VBScriptEngine()
   }
   public static ScriptEngine ScriptEngine { get; private set; }
   ...
}
```

You might want to read from the configuration file which scripting language you should enable within the application. This is one possible configuration schema:

```
<appSettings>
  <add key="language" value="vb" />
</appSettings>
```

Once you have an instance of your chosen scripting engine ready, you can run any valid JavaScript (or VBScript) code. There isn't much you can do in a real-world scenario until you're armed with these basic capabilities.

Add Scriptable Objects

All ClearScript engines expose a programmable interface through which you can add scriptable objects to the runtime environment. In particular, you use the method AddHostObject, like so:

```
ScriptEngine.AddHostObject("out", new SonoraOutput(settings));
ScriptEngine.AddHostObject("xml", new XmlFacade());
```

This method requires two parameters. The first parameter is the public name scripters will use to reference the object being published. The second parameter is just the object instance. Looking at the previous code snippet, in any JavaScript or VBScript you can use the name "out" to invoke any of the public methods available on the SonoraOutput interface. Here's an example in JavaScript that references what's shown in **Figure 1**:

```
var x = 4;
out.print(x + 1);
```

As you may know, it's a common practice in JavaScript to name members according to the camelCase convention. In .NET programming, the PascalCase convention is more common and also recommended. In my implementation of the SonoraOutput class, I deliberately opted to follow the JavaScript convention and called the method print instead of Print, as it would be the case in plain C# programming.

Based on my experience, there's not much more you need to know and understand to get started on ClearScript. Most of the time, you can configure a ClearScript environment within a host application with the primary purpose of making available application-specific objects. More often than not, these are tailor-made objects wrapped around existing business objects and made nicer to use from within a script environment.

The primary users of a ClearScript environment are usually not full-time developers. They're most likely people with some soft

Figure 3 **Code to Trigger Script Code**

```
public void Confirm()
{
  try
  {
    SonoraConsole.ScriptEngine.Execute(Command);
    OutputText = SonoraConsole.Output.ToString();
  }
  catch(Exception e)
  {
    OutputText = e.Message;
  }
}
```

development skills that would find it unnecessarily complex and annoying to deal with the full details of .NET classes. ClearScript lets you expose large chunks of the .NET Framework directly to JavaScript and VBScript. I opted to have tailor-made objects designed for extreme simplicity. Here's how you can publish in ClearScript a type instead of an object:

```
ScriptEngine.AddHostType("dt", typeof(DateTime));
```

When you reference a type, you're giving users the power to programmatically create instances of that type. For example, the previous line of code adds the power of the .NET DateTime object to the scripting environment. Now, the following JavaScript code becomes possible:

```
var date = new dt(1998, 5, 20);
date = date.AddDays(1000);
out.print(date)
```

From within JavaScript code, you're taking advantage of the full power of methods such as AddDays and AddHours. What if you want to get the difference between two dates? You can do something like this:

```
var date1 = new dt(1998, 5, 20);
var date2 = date1.AddDays(1000);
var span = date2.Subtract(date1);
out.print(span.Days)
```

The TimeSpan object is handled correctly and the expression span.Days just returns 1000. This is due to the dynamic nature of the JavaScript language, which dynamically determines the object named "span" exposes a member named Days. If you want to create a TimeSpan instance instead, you need to first let the engine know it's there.

To avoid exposing a million different types, ClearScript lets you host an entire assembly. This is one possible approach:

```
ScriptEngine.AddHostObject("dotnet",
  new HostTypeCollection("mscorlib", "System.Core"));
```

The keyword dotnet now becomes the key to access any types and static members within mscorlib and System.Core. Creating a new date object takes a bit longer, but in return you can explicitly work with TimeSpan objects:

```
var date1 = new dotnet.System.DateTime(1998, 5, 20);
var ts1 = new dotnet.System.TimeSpan(24, 0, 0);
var ts2 = ts1.Add(new dotnet.System.TimeSpan(24, 0, 0));
out.print(ts2.Days);
```

The JavaScript code snippet prints out the number 2, resulting from the sum of two distinct TimeSpan objects each counting for 24 hours. One thing that ClearScript doesn't work well with is operator overloading. That just doesn't exist. This means to sum up dates or timespans, you have to use methods such as Add or Subtract. It also supports reflection.

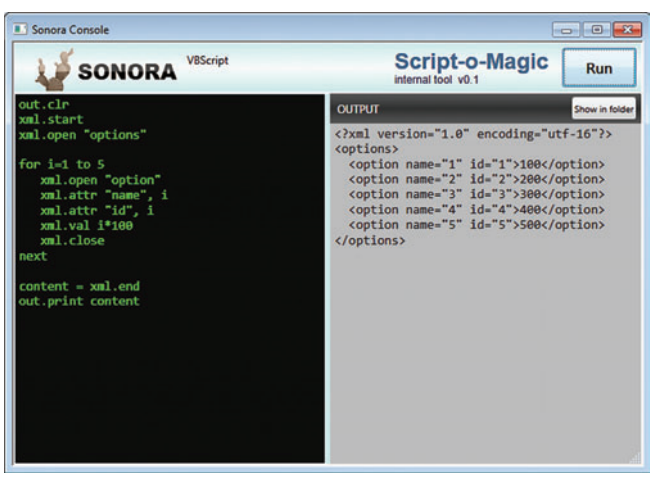


Figure 4 **Create an XML File via Script**

Generate the Output

The tool you see in **Figure 1** must be able to display some results to the user. By default, the SonoraOutput object added to the ClearScript engine just maintains an internal StringWriter object. All text processed by the print method is actually written to the underlying writer. The content of the writer is exposed to the outside world via the SonoraConsole class. This class is the sole point of contact between the ClearScript engine and the host application. The host application presenter just returns the content of the string writer through a property. That property is then bound to a TextBlock in the WPF UI. The method print writes to the UI via the string writer. The method clr clears the buffer and the UI.

Save to a Text File

My customer only needed to create text files, mostly CSV files. This is relatively easy to achieve. All I did was create a file method and pass it some text content directly. I could also let it grab anything already printed to the screen and saved in the internal buffer. The most problematic aspect to deal with when it comes to files is naming and location. To make scripting really fast, it has to be trivially easy to create and retrieve files. I managed to have two default folders—one for input and one for output. I also assume that all files are TXT. If no file name is specified, files assume a default name.

These assumptions are possibly too restrictive for some scenarios, but my project was simply a proof of concept for a tool to produce files, not store them. As **Figure 4** demonstrates, I could easily wrap the XmlWriter object into a nice component and create an XML file via script.

Wrapping Up

What's the point of creating an XML file via script? It's actually the same as having script capabilities in some enterprise applications. You need script because you want to automate tasks. In some cases, creating ad hoc text or XML files is just what you need. Perhaps you can run queries on SQL Server and import them to CSV, but that requires administrative access to the production database and, more important, the appropriate skills. I would have trouble myself using xp_cmdshell to grab text files out of SQL Server queries. For a developer, it might not be difficult to arrange some ad hoc, easy-to-use objects that are ready-made for scripting.

My customer loved this idea as much as I loved using ClearScript. He asked me to add a lot more objects to the dynamic environment. I ended up adding a layer of inversion of control to configure objects to be loaded at startup. He's also considering ClickOnce deployment across the company for when new tools are released. ■

Dino Esposito *is the co-author of "Microsoft .NET: Architecting Mobile Applications Solutions for the Enterprise" (Microsoft Press, 2014) and the upcoming "Programming ASP.NET MVC 5" (Microsoft Press, 2014). A technical evangelist for the .NET Framework and Android platforms at JetBrains and frequent speaker at industry events worldwide, Esposito shares his vision of software at software2cents.wordpress.com and on Twitter at twitter.com/despos.*

Thanks to the following technical experts for reviewing this article: *Microsoft ClearScript Team*

DOMAINS | MAIL | **HOSTING** | eCOMMERCE | SERVERS

NEW HOSTING

POPULAR APPS - NOW EVEN BETTER!

WordPress & Powerful App Platforms!

- Supports over 140 popular apps including WordPress, Drupal™, Joomla!™, TYPO3, and more
- Security & version upgrade notifications
- Trial version available for all applications
- App Expert Support
- 1&1 CDN powered by CloudFlare™ and 2 GB guaranteed RAM for peak performance

Powerful Tools

- PHP 5.5, Perl, Python, Ruby
- 1&1 Mobile Website Builder
- NetObjects Fusion® 2013 included

Successful Marketing

- Facebook® advertising credits
- Listing in business directories
- 1&1 Search Engine Optimization
- 1&1 E-Mail Marketing Manager

State-of-the-Art Technology

- Maximum availability (Geo-Redundancy)
- 300 Gbit/s network connection

All Inclusive

- **1 FREE** domain: .com, .net, .org, .biz, .info
- Unlimited Power: webspace, traffic, mail accounts, MySQL databases
- Secure e-mail addresses with virus and spam protection
- Linux or Windows operating system

COMPLETE PACKAGES FOR PROFESSIONALS

Starting at **$0.99** per month*

1 TRIAL 30 DAY MONEY BACK GUARANTEE

1 MONTH FLEXIBLE PAYMENT OPTIONS

1 CALL SPEAK WITH AN EXPERT 24/7

1 (877) 461-2631

1&1®

1and1.com

*Offer valid for a limited time only. The $0.99/month price reflects a 12-month pre-payment option for the 1&1 Basic Hosting Package. Regular price of $5.99/month after 12 months. Some features listed are only available with package upgrade. Visit www.1and1.com for full promotion details. Program and pricing specifications and availability subject to change without notice. 1&1 and the 1&1 logo are trademarks of 1&1 Internet, all other trademarks are the property of their respective owners.

KENNY KERR

DirectComposition: Transforms and Animation

DirectComposition visuals provide a lot more than just the offset and content properties I've illustrated in my last few columns. The visuals really come to life when you begin to affect them with transforms and animations. In both cases, the Windows composition engine is a sort of processor and it's up to you to calculate or construct the transform matrices, as well as the animation curves with cubic functions and sine waves. Thankfully, the Windows API provides the necessary support with a pair of very complementary APIs. Direct2D provides great support for defining transform matrices, making light work of describing rotation, scale, perspective, translation and much more. Likewise, the Windows Animation Manager frees you from having to be a wizard at mathematics, allowing you to describe animations using a rich library of animation transitions, a storyboard with key frames and loads more. It's when you combine the Windows Animation Manager, Direct2D and DirectComposition into a single application that you can really experience the sheer power at your fingertips.

In my previous column (msdn.microsoft.com/magazine/dn759437) I showed how the DirectComposition API can be used along with Direct2D to get the best of both retained-mode and immediate-mode graphics. The sample project included with that column illustrated this concept with a simple application that lets you create circles, move them around, and control their Z-order quite simply. In this column I want to show you how easy it is to add some powerful effects with transforms and animation. The first thing to realize is that DirectComposition provides overloads for many of its scalar properties. You might have noticed this if you've been following along over the last few months as I've explored the API. For example, a visual's offset can be set with the SetOffsetX and SetOffsetY methods, as follows:

```
ComPtr<IDCompositionVisual2> visual = ...

HR(visual->SetOffsetX(10.0f));
HR(visual->SetOffsetY(20.0f));
```

But the IDCompositionVisual interface IDCompositionVisual2 derives from also provides overloads of these methods that accept an animation object rather than a floating point value. This animation object materializes as the IDCompositionAnimation interface. For example, I could set a visual's offset with one or two animation objects, depending on whether I need to animate one or both axes, like so:

```
ComPtr<IDCompositionAnimation> animateX = ...
ComPtr<IDCompositionAnimation> animateY = ...

HR(visual->SetOffsetX(animateX.Get()));
HR(visual->SetOffsetY(animateY.Get()));
```

But the composition engine is capable of animating far more than just a visual's offset. Visuals also support 2D and 3D transformations. A visual's SetTransform method may be used to apply a 2D transform, given a scalar value:

```
D2D1_MATRIX_3X2_F matrix = ...

HR(visual->SetTransform(matrix));
```

Here, DirectComposition actually relies on the 3x2 matrix defined by the Direct2D API. You can do things like rotate, translate, scale and skew a visual. This affects the coordinate space the visual's content is projected in, but it's confined within two dimensional graphics with an X and Y axis.

Naturally, the IDCompositionVisual interface provides an overload of the SetTransform method, but it doesn't accept an animation object directly. You see, an animation object is responsible only for animating a single value over time. A matrix, by definition, consists of a number of values. You might want to animate any number of its members, depending on the effect you'd like to achieve. So, instead, the SetTransform overload accepts a transform object:

```
ComPtr<IDCompositionTransform> transform = ...

HR(visual->SetTransform(transform.Get()));
```

It's the transform object, specifically the various interfaces derived from IDCompositionTransform, that provides overloaded methods that accept either scalar values or animation objects. In this way, you might define a rotation matrix with an animated angle of rotation, but a fixed center point and axes. What you animate is, of course, up to you. Here's a simple example:

```
ComPtr<IDCompositionRotateTransform> transform = ...

HR(transform->SetCenterX(width / 2.0f));
HR(transform->SetCenterY(height / 2.0f));
HR(transform->SetAngle(animation.Get()));

HR(visual->SetTransform(transform.Get()));
```

The IDCompositionRotateTransform interface derives from IDCompositionTransform and represents a 2D transform that affects the rotation of a visual around the Z-axis. Here, I'm setting the center point to a fixed value, based on some width and height, and using an animation object to control the angle.

So that's the basic pattern. I've only described 2D transforms, but 3D transforms work in much the same way. Now let me illustrate more practically by showing you how you can take the sample project included with my previous column and transform and animate it in various ways with the help of Direct2D and the Windows Animation Manager.

Illustrating transforms and animation in a print magazine does begin to push the limits of what can easily be grasped by staring

at a static page. If you'd like to see it in action, check out my online course, where you can watch as it all comes to life (bit.ly/WhKQZT). To make the concepts a little more print-friendly, I'll take the liberty of tweaking the sample project from my previous column to use squares rather than circles. This should make the various transformations a little more obvious on paper. First, I'll replace the SampleWindow m_geometry member variable with a rectangle geometry:

```
ComPtr<ID2D1RectangleGeometry> m_geometry;
```

Then in the SampleWindow CreateFactoryAndGeometry method I'll get the Direct2D factory to create a rectangle geometry instead of the ellipse geometry:

```
D2D1_RECT_F const rectangle =
  RectF(0.0f, 0.0f, 100.0f, 100.0f);

HR(m_factory->CreateRectangleGeometry(
  rectangle,
  m_geometry.GetAddressOf()));
```

And that's all it takes. The rest of the app will simply use the geometry abstraction for rendering and hit testing as before. You can see the result in **Figure 1**.

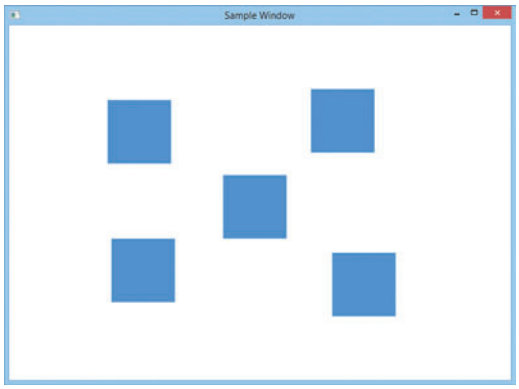


Figure 1 **Illustrating Transforms and Animation with Squares Instead of Circles**

Next, I'm going to add a simple message handler to respond to the WM_KEYDOWN message. In the SampleWindow MessageHandler method I'll add an if statement for this:

```
else if (WM_KEYDOWN == message)
{
  KeyDownHandler(wparam);
}
```

As usual, the handler will need the necessary error handling to recover from device loss. **Figure 2** provides the usual pattern of releasing the device resources and invalidating the window so the WM_PAINT message handler can rebuild the device stack. I'm also limiting the handler to the Enter key to avoid confusion with the Control key, which is used when adding shapes.

At this point, I'm ready to start experimenting with transforms and animations. Let's begin simply with a basic 2D rotation transform. First, I need to determine the center point that will represent the Z axis, or the point around which to rotate. Because DirectComposition expects physical pixel coordinates, I can simply use the GetClientRect function here:

```
RECT bounds {};
VERIFY(GetClientRect(m_window, &bounds));
```

I can then derive the center point of the window's client area, as follows:

```
D2D1_POINT_2F center
{
  bounds.right / 2.0f,
  bounds.bottom / 2.0f
};
```

I can also rely on the Direct2D matrix helper functions to construct a matrix describing a 2D rotation transform of 30 degrees:

```
D2D1_MATRIX_3X2_F const matrix =
  Matrix3x2F::Rotation(30.0f, center);
```

And then I can simply set a visual's transform property and commit the change to the visual tree. I'll just apply this change to the root visual for simplicity:

```
HR(m_rootVisual->SetTransform(matrix));
HR(m_device->Commit());
```

You can, of course, apply any number of changes to any number of visuals and the composition engine will take care of coordinating it all effortlessly. You can see the results of this simple 2D transform in **Figure 3**. You might notice some aliasing. Even though Direct2D defaults to anti-aliasing, it assumes the drawing will appear in the coordinate space in which it was rendered. The composition engine doesn't have any knowledge of the geometry the composition surface was rendered with, so it has no way of correcting this. In any case, once animation is added to the mix, the aliasing will be short-lived and hard to spot.

To add animation to this transformation, I need to switch the matrix structure out for a composition transform. I'll replace both the D2D1_POINT_2F and D2D1_MATRIX_3X2_F structures with a single rotate transform. First, I need to create the rotate transform using the composition device:

```
ComPtr<IDCompositionRotateTransform> transform;
HR(m_device->CreateRotateTransform(transform.GetAddressOf()));
```

Keep in mind that even such seemingly simple objects must be discarded if and when the device is lost and recreated. I can then set the center point and angle using interface methods rather than a Direct2D matrix structure:

```
HR(transform->SetCenterX(bounds.right / 2.0f));
HR(transform->SetCenterY(bounds.bottom / 2.0f));
HR(transform->SetAngle(30.0f));
```

And the compiler picks the appropriate overload to handle the composition transform:

```
HR(m_rootVisual->SetTransform(transform.Get()));
```

Running this produces the same effect as in **Figure 3**, because I haven't yet added any animation. Creating an animation object is simple enough:

```
ComPtr<IDCompositionAnimation> animation;
HR(m_device->CreateAnimation(animation.GetAddressOf()));
```

Figure 2 **Scaffolding for Device Loss Recovery**

```
void KeyDownHandler(WPARAM const wparam)
{
  try
  {
    if (wparam != VK_RETURN)
    {
      return;
    }

    // Do stuff!
  }
  catch (ComException const & e)
  {
    TRACE(L"KeyDownHandler failed 0x%X\n",
      e.result);

    ReleaseDeviceResources();

    VERIFY(InvalidateRect(m_window,
                          nullptr,
                          false));
  }
}
```

I can then use this animation object instead of the constant value when setting the angle:

```
HR(transform->SetAngle(animation.Get()));
```

It gets more interesting when you try to configure the animation. Simple animations are relatively straightforward. As I mentioned earlier, animations are described with cubic functions and sine waves. I can animate this rotation angle with a linear transition, where the value progresses from 0 to 360 over 1 second by adding a cubic function:

```
float duration = 1.0f;

HR(animation->AddCubic(0.0,
                       0.0f,
                       360.0f / duration,
                       0.0f,
                       0.0f));
```

The third parameter of the AddCubic method indicates the linear coefficient, so that makes sense. If I left it there, the visual would rotate 360 degrees every second for eternity. I can decide to bring the animation to an end once it reaches 360 degrees, as follows:

```
HR(animation->End(duration, 360.0f));
```

The first parameter of the End method indicates the offset from the beginning of the animation, regardless of the value of the animation function. The second parameter is the final value of the animation. Keep this in mind because the animation will "snap" to that value if it doesn't line up with the animation curve, and that would produce a jarring visual effect.

Such linear animations are easy enough to reason about, but more complex animations can become exceedingly complicated. That's where the Windows Animation Manager comes in. Rather than having to call the various methods of IDCompositionAnimation to add sinusoidal and cubic polynomial segments, repeated segments, and more, I can construct an animation storyboard with the Windows Animation Manager and its rich library of transitions. When that's done, I can use the resulting animation variable to populate the composition animation. This involves a bit more code, but the benefit is a great deal more power and control over your app's animation. First, I need to create the animation manager itself:

```
ComPtr<IUIAnimationManager2> manager;

HR(CoCreateInstance(__uuidof(UIAnimationManager2),
  nullptr, CLSCTX_INPROC, __uuidof(manager),
  reinterpret_cast<void **>(manager.GetAddressOf())));
```

Yes, the Windows Animation Manager relies on COM activation, so be sure to call CoInitializeEx or RoInitialize to initialize the runtime. I also need to create the transition library:

```
ComPtr<IUIAnimationTransitionLibrary2> library;

HR(CoCreateInstance(__uuidof(UIAnimationTransitionLibrary2),
  nullptr, CLSCTX_INPROC, __uuidof(library),
  reinterpret_cast<void **>(library.GetAddressOf())));
```

Typically, applications will hold on to these two objects for their lifetime as they're needed for continuous animation and, specifically, for velocity matching. Next, I need to create an animation storyboard:

```
ComPtr<IUIAnimationStoryboard2> storyboard;
HR(manager->CreateStoryboard(storyboard.GetAddressOf()));
```

The storyboard is what associates transitions with animation variables and defines their relative schedule over time. The storyboard is able to aggregate various transitions applied to different animation variables; it ensures they remain synchronized; and it's the storyboard that's scheduled as a whole. Of course, you can create multiple storyboards to schedule independent animations. Now, I need to ask the animation manager to create an animation variable:

```
ComPtr<IUIAnimationVariable2> variable;

HR(manager->CreateAnimationVariable(
  0.0, // initial value
  variable.GetAddressOf()));
```

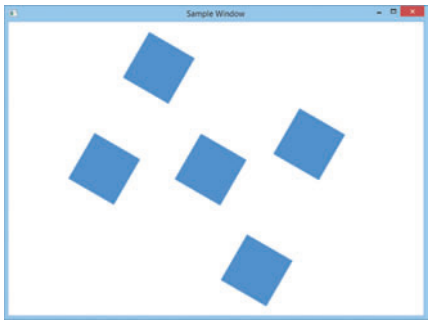


Figure 3 **A Simple 2D Transform**

Once a storyboard is scheduled, the animation manager is responsible for keeping the variable up-to-date so the app can request the effective value at any time. In this case, I'm simply going to use the animation variable to populate the composition animation with segments and then discard it. Now I can use the mighty transition library to create an interesting transition effect for the animation variable:

```
ComPtr<IUIAnimationTransition2> transition;

HR(library->CreateAccelerateDecelerateTransition(
  1.0,   // duration
  360.0, // final value
  0.7,   // acceleration ratio
  0.3,   // deceleration ratio
  transition.GetAddressOf()));
```

The transition will cause the animation variable to speed up and then slow down over the given duration until it comes to a rest at its final value. The ratios are used to affect how relatively quickly the variable will accelerate and then decelerate. Keep in mind that the ratios combined can't exceed a value of one. With the animation variable and transition ready to go, I can add them to the storyboard:

```
HR(storyboard->AddTransition(variable.Get(),
                             transition.Get()));
```

The storyboard is now ready to be scheduled:

```
HR(storyboard->Schedule(0.0));
```

The single parameter of the Schedule method tells the scheduler the present animation time. This is more useful for coordinating animation and coordinating with the composition engine's refresh rate, but it will do for now. At this point, the animation variable is primed and I can ask it to populate the composition animation with curves:

```
HR(variable->GetCurve(animation.Get()));
```

In this case, it's as if I called the composition animation, as follows:

```
HR(animation->AddCubic(0.0, 0.0f, 0.0f, 514.2f, 0.0f));
HR(animation->AddCubic(0.7, 252.0f, 720.0f, -1200.0f, 0.0f));
HR(animation->End(1.0, 360.0f));
```

That's certainly a lot less code than it took with the Windows Animation Manager, but making sense of that math isn't that simple. The Windows Animation Manager also offers the ability to coordinate and smoothly transition running animations, something that would be exceedingly difficult to do manually. ■

Kenny Kerr *is a computer programmer based in Canada, as well as an author for Pluralsight and a Microsoft MVP. He blogs at kennykerr.ca and you can follow him on Twitter at twitter.com/kennykerr.*

Thanks to the following Microsoft technical experts for reviewing this article: *Leonardo Blanco and James Clarke*

JULIE LERMAN

Git: It's Just Data!

Source control in a data column? Ahh, but when that source control is just one big database, it's an invitation to some data-geeky fun. To be clear, you should know right up front I'm not writing about Git this month because I'm an expert. In fact, I'm writing about Git because I have struggled with it. My somewhat anemic GitHub profile is testament to that fact. When the subject of Git comes up, I tend to change the subject for fear of my Gitlessness being discovered—as most of my developer friends interact with it without batting an eyelash.

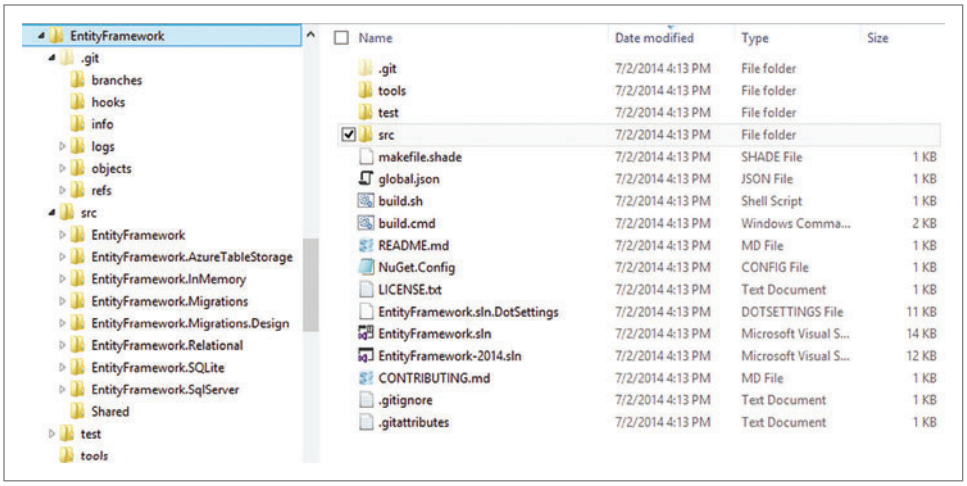


Figure 1 **The Solution Folder Created by Cloning the Repository, Including the Repository Itself in the .git Folder**

While talking about my Gitless life recently with friend and local Ruby developer, Alan Peabody (github.com/alanpeabody), at a local hacker happy hour, he said to me, "But Julie, it's just data." Data? "Oooh, I love data! Tell me more!" Peabody described how Git relies on a database filled with key/value pairs, and then suggested that, rather than trying to use the available UI tools or what's known as the "porcelain" commands for working with Git, I explore and play with the lower-level "plumbing" commands and APIs.

So Git has a database and accessible plumbing! Peabody's advice absolutely inspired me. I've now spent some time exploring how Git works at a low level, how to read and write data representing repository activity, and how the different object types in Git relate to one another. I'm still far from expert but feel much more in control of my own activity on Git. More important, I'm having fun playing with Git and no longer fear it as a source of black magic.

If this path appeals to you, I recommend not missing the "Git Basics" and "Git Internals" chapters of the online "Pro Git" book by Scott Chacon (Apress, 2009) and available at git-scm.com/book.

I'm sure the Git database was pointed out to me before, but only in passing and while I was already overwhelmed by the learning curve. Now I plan to have a little fun with it. For the rest of this article, I'll simply interact with the database and some code that it's tracking and see how it affects the database that represents the repository.

Code download available at msdn.microsoft.com/magazine/msdnmag0914.

Rather than starting with a new, empty repository, I'll use an existing repository I've already been playing with: the very early work on Entity Framework 7 hosted at GitHub.com/aspnet/EntityFramework. Keep in mind this is a rapidly changing project at the time I'm writing this column, so the files I work with here might change.

I'll execute the commands from Windows PowerShell, leveraging posh-git, which enhances the command-line experience with additional status information, color-coding and tab completion. You can find instructions for setting this up in the download that accompanies this article.

Getting a Repository and Examining Git Assets

The first step is to clone the existing repository, starting in my existing github folder and using the git clone command:

```
D:\User Documents\github> git clone git://github.com/aspnet/EntityFramework
```

By itself, this first step elevated my Git skills! Git creates a new folder in my starting directory using the name of the repository, so I get D:\User Documents\github\EntityFramework. During the cloning process, if you open the EntityFramework folder in File Explorer as soon as it's created, you'll see that a .git subfolder is created first. This is the repository and is what we call the git database. And Git is using this data to construct the source in what's referred to as the working directory, that is, the rest of the EntityFramework folder.

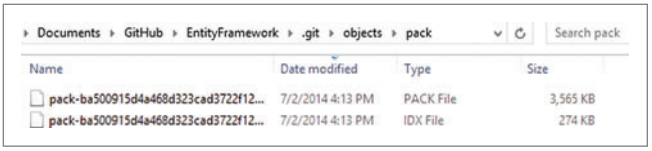


Figure 2 **You Won't See Objects at First—They're All Compressed into a PACK File**

Figure 3 **Windows PowerShell with posh-git Activated**

Once the operation is complete, the new folder, EntityFramework, looks much like any other Visual Studio solution folder except for one thing: the .git folder that is, itself, a complete copy of the cloned repository, including all branches. Notice its file structure in **Figure 1**. In Git, a branch is just a pointer to a commit and a commit is a pointer to a snapshot of your working directory. Although a Git repository's "main" branch is called master by default, that's not a requirement. The EF team set their default branch to point to a branch they named dev. Using branches, you can modify and test code safely, before merging changes into other branches.

The rest of the EntityFramework folder contents represent the working directory. At some point, you tell Git to keep track of files you've added, changed or deleted in your working directory. This is referred to as "staging"—that is, the changes to these files are staged and ready to be committed. Staged changes are stored in the database and will remain there after you've committed them. Eventually, you push those changes up to the server. I won't be dealing with that step in this article because my focus is on exploring the database activity. There are many other concepts, such as trees and forking, references and headers that I'll also ignore for this journey.

So what and where is this database? Is it a relational database like SQL Server or SQL CE? Nope. It's a collection of files that represent Git objects. Each file is named with a hash and contains hashed contents. Remember that the database is a set of key/value pairs. The filename is the key that represents that object in the database, and its contents are the value. The collection of these objects comprises the database that represents the repository. Every key/value pair is stored in the .git/objects folder, and a master list of them is stored in a file called index. There are other objects that keep track of the different branches. It's possible that many files are just duplicated and never edited. Git has a way to point to those files rather than maintain separate copies, which keeps the Git database from bloating.

But there are no object files to be seen yet, because, initially, everything is compressed into a pack file along with its IDX counterpart. These are in the PACK subfolder as shown in **Figure 2**.

What Does Git Know About My Code?

Before I start editing, I want to take a look at what Git knows about my working directory and the repository. This means going back to Windows PowerShell and enabling posh-git to enhance my Git command-line experience. I start by changing directory to the EntityFramework folder (*cd EntityFramework*, just like the good old DOS days). Posh-git will see the .git subfolder and engage in the Windows PowerShell environment. **Figure 3** shows my Windows PowerShell window now has the title posh~git and the prompt displays a status in yellow brackets—those are posh-git features. Currently, it indicates that my working directory is using the branch called dev, the main branch of the EF7 repository. When I cloned the repository, by default Git "checked out" this branch and that's the version it put in my working directory. That's really all that checking out means to Git. It builds the working directory from the designated branch. A fun exercise is to delete everything but the .git folder in File Explorer, then type "git checkout dev" in Windows PowerShell to see the directory contents get completely recreated. Unplug from the Internet if you want proof that it's not coming from the server.

I can ask Git what it knows about my working directory with the git "status" command. All git commands start by addressing git, so type:

```
git status
```

It responds with the message:

```
On branch dev
nothing to commit, working directory clean
```

That makes sense. I'm starting with a clean slate.

See How Git Responds to Editing Your Working Directory Files

Now it's time to start banging on things to see this status change.

I'll edit one of my favorite EF classes: DbContext.cs (which is in src\EntityFramework). I won't even bother with Visual Studio. You can just use NotePad++ or your favorite text editor.

I added a comment at the top:

```
// Julie was here
```

and saved.

When I run git status again, the response is now more interesting, as **Figure 4** shows.

Git sees that the file (noted in an unfortunately hard-to-read red font) has changed, but it's not staged. In other words, Git isn't tracking it yet, but has compared the working directory files to its database to make that determination. Notice also the new prompt status for dev is +0 ~1 -0. Status displayed in red reflects the working directory, here indicating 0 new files, 1 modified file and no deleted files.

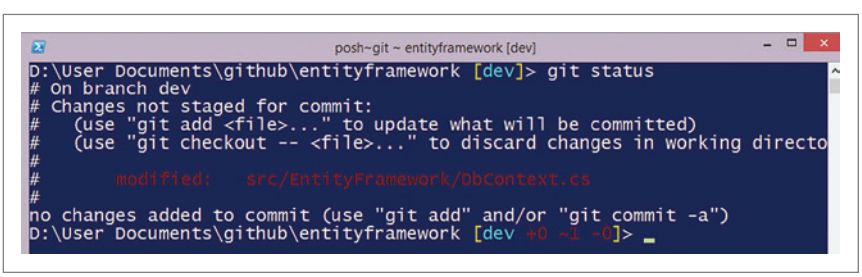


Figure 4 **Status After Modifying a File in the Working Directory (Red Font Indicates Working Directory Status)**

Figure 5 **An Object Being Tracked by Git in Its Objects Folder**

What about the database? If you look in File Explorer, you'll see by its timestamp and size that the index file hasn't changed. Nothing has changed in the objects folder, either. The Git database is unaware of the change in my working directory.

Dear Git, Please Track My File Now

Git won't track files until you tell it to. With typical porcelain commands, you just tell Git to add and it figures out which files to pull in. However, I'm going to use a plumbing command instead of the more common commands devs use with Git so I can be very explicit about each step I want to take. Note that the file path I type is case-sensitive:

```
git update-index --add src\EntityFramework\DbContext.cs
```

In response, the prompt's status changes. It still says +0 ~1 -0, but the font is green now, not red, indicating this is the status of the index. Git is now tracking 0 new objects, 1 modified object and 0 deleted objects, which are ready to be committed.

What about the Git index file and the objects folder?

> Git won't track files until you tell it to.

The timestamp of the index file in the .git directory has changed. The file now has a notation that the object file that represents the DbContext.cs class has changed. This is how the status knows one modified file is being tracked. For you Entity Framework coders, does this sound familiar? The index is akin to the EF DbContext, which tracks changes to your entities! DbContext knows when an object instance has been added or modified or deleted. The index file is similar in this way.

But now you can see the object, too. Browse to .git/objects and you'll see a new folder. If you edited in Visual Studio, you might see more, for example, if the project file changed. But I edited in Notepad++ and there's just one new folder, named ae. In the folder is a file with a name that's a hash, as shown in **Figure 5**.

This object is part of my database and overrides the object representing the DbContext.cs file that's cached in the PACK file. That new object file contains a hash of the contents of DbContext.cs, including my change.

I can't read it myself, but Git can. I'll ask Git to display its entire contents with the cat-file command:

```
git cat-file -p aeb6db24b9de85b7b7cb833379387f1754caa146
```

The -p parameter requests a prettified listing of the text. It's followed by the name of the object to list, which is a combination of the folder name (ae) and the file name. Git has shortcuts for expressing this object name.

A more interesting view is one that highlights the change. I can ask Git to show me what has changed in this branch (remember its name is "dev") with:

```
git diff dev
```

The response, to which I've added line numbers for clarity, is shown in **Figure 6**.

This text is formatted using colored fonts that make it easy to distinguish the different information. Notice that on line 8, there's a dash (which is red), indicating I deleted a [blank] line. Line 9 begins with a plus sign, indicating a new line, and it's displayed in a green font. If I added more objects to the index—for modified or new files—they'd be listed here, as well.

Even though the index file is binary, I can also explore it. The git ls-files command is handy for listing all of the files represented by repository objects in the index file. Unfortunately, ls-files lists a combination of the cached index and the working directory, showing every file in my solution, so I have to dig through it to find the files in which I'm interested. There's another git command, grep, that's used for filtering. You can combine that with ls-files to filter the files. I'll add grep (which is case-sensitive) and also ask ls-files to display the object name by using its -s (stage) parameter:

```
git ls-files -s |grep DbContext.cs
```

Including the stage parameter forces the output to contain a file-mode indicator (100664 means it's a non-executable group-writable file) and the hash filename of the object with which it's associated:

```
100644 aeb6db24b9de85b7b7cb833379387f1754caa146 0
src/EntityFramework/DbContext.cs
```

There's much more to ls-files, but this is what I'm interested in at the moment—the fact that I can see how Git is mapping the object file to my working directory's DbContext.cs file.

What Does Committing Do to the Git Database?

Next, let's see a bit of the effects of pushing objects from staged to committed. Remember that this commit happens only on your computer. Nothing gets committed directly to the original repository on the server. The git commit command requires you add a note using the -m parameter. Keep in mind you're committing all staged changes. I've made just one here:

```
git commit -m "Edited DbContext.cs"
```

```
1  D:\User Documents\github\entityframework [dev +0 ~1 -0]> git diff dev
2  diff --git a/src/EntityFramework/DbContext.cs b/src/EntityFramework/DbContext.cs
3  index 7d0c554..aeb6db2 100644
4  --- a/src/EntityFramework/DbContext.cs
5  +++ b/src/EntityFramework/DbContext.cs
6  @@ -1,6 +1,6 @@
7   // Copyright (c) Microsoft Open Technologies, Inc. All rights reserved.
8   // Licensed under the Apache License, Version 2.0. See License.txt in the project roo
9  -
10 +// Julie was here
11  using System;
12  using System.Threading;
13  using System.Threading.Tasks;
```

Figure 6 **Response to git diff Command**

Git responds by telling me the newly generated Git database name of the object (2291c49) that contains the commit information and what was committed. The insertion note is about what happened in the file—I added something:

```
[dev 2291c49] Edited DbContext.cs
1 file changed, 1 insertion(+)
D:\User Documents\github\
  entityframework [dev]>
```

Notice the prompt is clean. The status now reflects everything that's happened since the last commit, which is nothing. I'm back to a clean slate.

What has happened in my database in response to the commit? I can tell that the index file wasn't updated because its timestamp hasn't changed.

But there are four new folders in the objects directory, each containing its own hash object. I use cat-file to see what's in those files. The first is a listing, similar to ls-files, of all the files and folders in the src\EntityFramework folder, along with their object names. The second is a list of all the objects in the src folder—that is, a list of the subfolders. These, by the way, are "trees," not "folders," in git-speak. The third object contains a list of all the files and folders in the root EntityFramework folder. The final object contains data that will be needed in order to sync to the server—my committer and author identity (my e-mail address), and one object for "tree" and another for "parent." Interestingly, this final object is the same object that was relayed in the response to the commit. It's in a folder named 22 and its file name begins with 91c49.

All of these objects will be used for pushing up to the server repository when the time comes.

Have I Pwnd Git?

In a way, I've taken ownership of Git by exploring things at a low level. I love exploring cause and effect as well as digging into plumbing. I've definitely lost my fear of Git's magic, and playing around with it in this way has also serendipitously given me an understanding of the command language and the beauty of working with Git at a low level. And it's probably the most time I've ever spent in Windows PowerShell, too.

I have friends who are Git fanatics—like Cori Drew who jumped on Skype with me a number of times when I got confused by what I was seeing—and many more who use Git as though it's the very air they breathe. They tried to show me how Git works, but I initially gave up on it. This exercise definitely empowered me to learn more and benefit from it, and to see how much more there is to learn. But it's only my fascination with data and with banging on things to see how they react that got me to this point. And for that I am grateful for Alan Peabody's inspiring suggestion.

Update: A week after I originally finished writing this article, I found myself using Git on two separate solo projects, branching and merging like an almost-pro! My own plunge was a great success and I hope yours will go just as well. ■

Julie Lerman *is a Microsoft MVP, .NET mentor and consultant who lives in the hills of Vermont. You can find her presenting on data access and other .NET topics at user groups and conferences around the world. She blogs at thedatafarm.com/blog and is the author of "Programming Entity Framework" (2010), as well as a Code First edition (2011) and a DbContext edition (2012), all from O'Reilly Media. Follow her on Twitter at twitter.com/julielerman and see her Pluralsight courses at juliel.me/PS-Videos.*

Thanks to the following technical experts for reviewing this article: *Cori Drew and Alan Peabody*

Soup to Nuts: From Raw Hardware to Cloud-Enabled Device

Bruno Terkaly and Steven Edouard

There's a new gold rush happening and it's not about precious metals. It's about building innovative devices for consumers and connecting them to the cloud. New Microsoft CEO Satya Nadella said last March that devices are "really uninteresting without the cloud."

This makes sense. If the device is collecting information through sensors or cameras, how and where will you analyze this data? After all, these low-end devices such as the Raspberry Pi have limited compute power and storage space. The cloud fills this void and goes further by helping with security, device management and so on. So this month, we want to roll up our sleeves from both a hardware and software perspective and learn what it takes to truly embrace this new computing paradigm.

The SmartDoor is the Internet of Things (IoT) product we'll build from the ground up. We will eventually attach a doorbell, a camera and a Raspberry Pi computing device (see **Figure 1**). The idea is that someone arrives at your doorstep and rings your doorbell. Once they press the doorbell, the device automatically takes a photo. Then this photo is forwarded to a mobile device as a push notification message (see **Figure 2**). The message has a hyperlink to the image of the person standing at your doorstep. This invention was inspired by the need to know who is ringing your doorbell, even if you're not home.

We'll begin by illustrating exactly how to purchase and assemble the raw hardware. Once we've built the hardware, we'll build software that runs on the device itself, as well as the software running in Microsoft Azure. This and future articles, though, will focus primarily on the requisite software.

We'll need the code to be able to take a picture and upload it to Azure. From there, it will need to send a push notification to the appropriate mobile devices. This will indicate that someone has pressed the doorbell, and a photo of that person is available. Because of the ambitious nature of this project, we'll provide a number of blog posts to support the work and provide the needed details.

This article discusses:

- How to control devices using Microsoft Azure Mobile Services as the platform
- Assembling the configuring hardware components
- Integrating a Raspberry Pi device with common hardware

Technologies discussed:

Microsoft Azure, Mono, C#, Microsoft .NET Framework, Node.js, JavaScript

Easy as Pi

We'll address the code that runs on the Raspberry Pi device itself and some of the code that will run in the cloud—specifically within Azure Mobile Services. On the Raspberry Pi, we'll install and run Mono, which is a free, open source project led by Xamarin that provides a Microsoft .NET Framework-compatible set of tools including a C# compiler and a CLR.

The beauty of Mono is it lets you write .NET code compiled on Windows and run it unchanged on the Raspberry Pi. In our case, we'll take a photo and upload it to Azure storage. In future articles, we'll expand the code base to support the ability to send push notifications to a Windows Phone. To do this, we'll have to leverage Push Notifications in the Azure Service Bus and write a Windows Store or Windows Phone app.

Getting the hardware together is straightforward because almost all of it is available in a single Raspberry Pi kit called the Canakit. For more details, go to canakit.com. The kit contains a few useful things to get to get our IoT device built, including a system board, A/C adapter, general purpose input/output (GPIO) breakout, breadboard, breadboard wires, assorted resisters and LEDs. You'll also need an external monitor and keyboard and mouse so you can configure and test the Raspberry Pi. Besides the Canakit, you'll need two more things—a regular doorbell and a Raspberry Pi-compatible camera, both of which are easy to find.

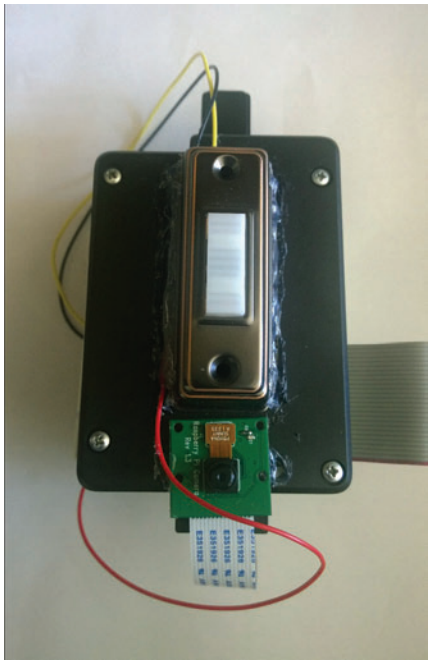


Figure 1 **This Project Combines a Doorbell, Camera and a Raspberry Pi Device**

While the pile of hardware in **Figure 3** might look intimidating, it's actually quite simple to put together. If you can build a jigsaw puzzle, you can build a device like the SmartDoor. First, plug the camera module into the Raspberry Pi board. Take the camera out of the bag and attach the flex cable to the Raspberry Pi board. Go to bit.ly/1rk3vzk for more details on sending photos to the cloud.

We'll present full installation of the doorbell itself in a later article, where we will solder it to the Raspberry Pi. For now, we'll attach a breadboard, which is a test version of an electrical circuit plugged into a Raspberry Pi. We can use jumper cables and the breadboard to simulate the doorbell.

One of the core components is the GPIO breadboard. The GPIO is a hardware circuit with 26 separate pins (numbered left to right) that let you expand the system to interact with other devices. Generally, these pins let you connect such things as sensors, actuators, LEDs and so on. There are different types of pins, though. For example, there are two pins providing a power source for connected devices, specifically a 3.3 volt and a 5 volt. There's also a 0 volt pin that acts as a ground, which is necessary to define a circuit. Pins labeled GPIO act as a simple on/off switch. If you wish to perform serial communications, you'll find TX and RX pins (RS-232) for transmitting and receiving data. The software that runs on the device will need to communicate with these pins.

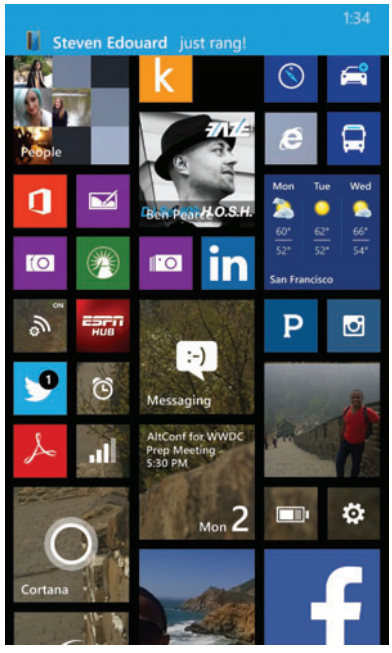


Figure 2 **The Windows Phone Receiving the Push Notification**

One of the complexities in dealing with these pins is there's a mismatch between the physical pins versus the logical pins with which the software communicates. This has to do with the Broadcom chips in the Raspberry Pi device. This means if a signal goes to the pin GPIO 0, your software will actually need to read pin GPIO 17. It can be illogical, so be careful.

There are other crucial hardware components. Resistors play a key role controlling current flow and lowering voltage levels, so you don't fry the Raspberry Pi or any other devices connected to it. You'll also need single-core wire to connect the GPIO unit to the Raspberry PI.

Testing, Testing

Consequently, we'll need to verify that we've built the hardware correctly. The easiest way to test the Raspberry Pi is to simply plug in the external monitor, keyboard, optional mouse and power supply. Once that's done, plug in the power and the boot sequence for Linux should appear. The default username will be "pi" and the password will be "raspberry." Go to bit.ly/1k8xBFn for more details on setting up a Raspberry Pi.

The Raspberry Pi needs additional software such as Mono to be able to run C# code. Before we can do anything on the device, we'll need to update its Raspbian OS. Luckily, Pi comes with a preinstalled version. All we need to do is issue a couple of commands in a console window to update the OS with the latest revision. Then we can install the Mono runtime from Xamarin and some trusted root certificates. These will let the device make HTTPS requests. All these steps are explained in the blog post at bit.ly/Unehrr.

The Software Side

There are two tiers we need to code against, the client tier (the Raspberry Pi device itself) and the service tier (Azure Storage and Azure Mobile Services). There are a variety of languages and back-end services from which to choose. We'll run C# on the client and Node.js/JavaScript on the server. This can be a controversial choice because some folks feel two different languages complicate the implementation.

Our goal was to show diversity. In the context of Node.js, we believe the numerous libraries and packages found at npmjs.org could potentially make your server code much easier. Node.js often results in a much smaller code base because the libraries are

Figure 3 **The Raspberry Pi Canakit (Does Not Include the Camera)**

so powerful. Less code in a second language is better than lots of code in the same language. The diagram in **Figure 4** highlights the expected workflow, which boils down to three steps:

1. The client requests a Shared Access Signature (SAS) token, which is presented as a URL.
2. The Node.js application running within Azure Mobile Services will return a SAS URL.
3. The client will use the SAS URL to upload the photo into Azure Storage as a blob.

We hosted our Node.js application within Azure Mobile Services. This provides a lot of pre-built support for mobile device clients, such as push notifications, CRUD storage integration, identity and the ability to build a custom API.

Focus on Service

We'll start by building our service, because it will develop the client if there's no service tier against which to code. There would be no way to actually run and test the client unless you have a fully functional back-end service. Azure Mobile Services will host our service API.

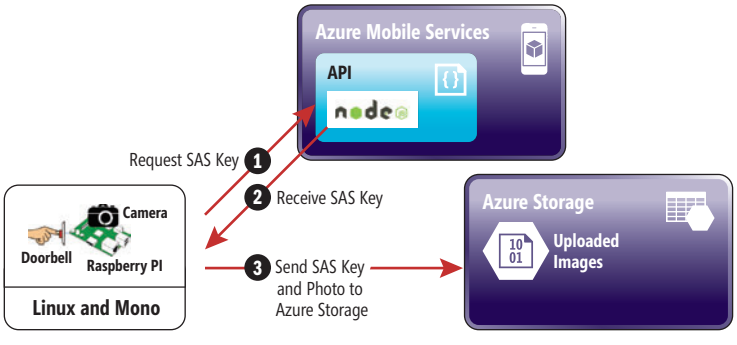


Figure 4 **High-Level Architectural Diagram of the SmartDoor Project**

This custom API will be based on Node.js and the Express Web framework. Our client (Raspberry Pi) will use those services to request a SAS. Then the service will use that token to upload the photo it takes to Azure Storage. Using a SaS URL helps protect our Azure Storage account, because we won't need to store the account credentials in the Raspberry Pi.

We addressed additional security issues in a past article (msdn.microsoft.com/magazine/dn574801). SAS tokens are also good for another reason—they free the middle tier (Node.js API) from having to broker the transfer of files to storage from the client.

Provisioning Mobile and Storage

Provisioning Azure Mobile Services consists of just a few clicks in the Azure portal. The same is true for provisioning Azure Storage. As with anything that's provisioned, you'll need to think about URLs and the naming conventions you want to use for your cloud-based services. You also should consider which of the several global datacenters you'd like to use, making sure your storage account is in the same datacenter as Azure Mobile Services, so as to minimize latency and avoid data transfer costs. You can find more detailed steps for provisioning your Azure Mobile Services and Azure Storage account at bit.ly/WyQird.

Shared Access Signatures

As shown in **Figure 4**, the Raspberry Pi client will request an SAS token from Azure Mobile Services. It will do so by issuing a get request from an API we define within Azure Mobile Services. SAS tokens free the middle tier (Node.js API) from having to broker the transfer of files to storage from the client.

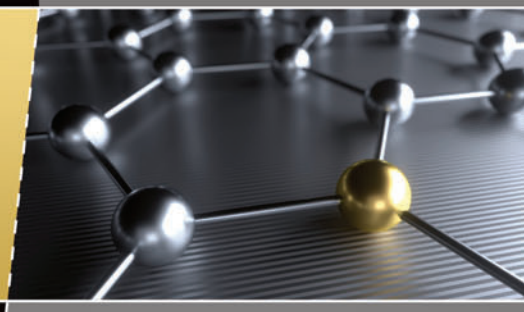
> If you can build a jigsaw puzzle, you can build a device like the SmartDoor.

The code in **Figure 5** is an excerpt of the Node.js running within Azure Mobile Services as an API service endpoint. The purpose is to respond to requests from Raspberry Pi clients that need an SAS. The Raspberry Pi will issue a "get" request against the endpoint hosted in Azure Mobile Services, passing in the application key, which is an identifier that uniquely represents our mobile service.

The application key represents a secure way to communicate with the specific Azure Mobile Service. The service will obtain this key from the portal, which the code running on the Raspberry Pi will use, so you should keep it handy. Then you can paste it into the client code. The client receives an SAS encapsulated within a URL the client can use in a later operation to upload a photo.

Now we'll focus on some of the code that represents the server side Node.js application. There's a powerful Node.js SDK tailored for Azure available at bit.ly/Unnikj. This code base provides a lot of support for other parts of the Azure platform, such as working with tables, queues, topics, notification hubs, core management and so on. You can find a tutorial on integrating Node.js and Azure Blob Storage at bit.ly/1nBEvBa.

Switch to Amyuni PDF



Create & Edit PDFs in .Net - ActiveX - WinRT

- Edit, process and print PDF 1.7 documents
- Create, fill-out and annotate PDF forms
- Fast and lightweight 32- and 64-bit components for .Net and ActiveX/COM
- New WinRT Component enables publishing C#, C++CX or Javascript apps to Windows Store
- New Postscript/EPS to PDF conversion module

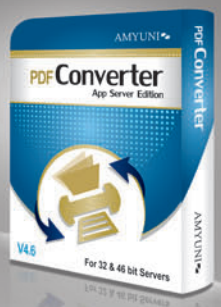
Complete Suite of Accurate PDF Components

- All your PDF processing, conversion and editing in a single package
- Combines Amyuni PDF Converter and PDF Creator for easy licensing, integration and deployment
- Includes our Microsoft WHQL certified PDF Converter printer driver
- Export PDF documents into other formats such as JPeg, PNG, XAML or HTML5
- Import and Export XPS files using any programming environment

Advanced HTML to PDF & XAML

- Direct conversion of HTML files into PDF and XAML without the use of a web browser or a printer driver
- Easy Integration and deployment within developer's applications
- WebkitPDF is based on the Webkit Open Source library and Amyuni PDF Creator

High Performance PDF Printer For Desktops and Servers



- Our high-performance printer driver optimized for Web, Application and Print Servers. Print to PDF in a fraction of the time needed with other tools. WHQL tested for Windows 32 and 64-bit including Windows Server 2012 R2 and Windows 8.1
- Standard PDF features included with a number of unique features. Interface with any .Net or ActiveX programming language
- Easy licensing and deployment to fit system administrator's requirements

CERTIFIED FOR Windows Server 2008 · Windows 7 · Windows 8 · Windows Server 2012 Certified

AMYUNI

All development tools available at

www.amyuni.com

USA and Canada
Toll Free: 1866 926 9864
Support: 514 868 9227
sales@amyuni.com

Europe
UK: 0800-015-4682
Germany: 0800-183-0923
France: 0800-911-248

Figure 5 **Node.js Code for the Microsoft Azure Mobile Services API**

```
exports.get = function(request, response) {

  // These are part of "App Settings" in the configuration section
  // of your service. Shouldn't be hardcoded into this Node.js script.
  containerName = request.service.config.appSettings.PhotoContainerName;
    accountName = request.service.config.appSettings.AccountName;
    accountKey = request.service.config.appSettings.AccountKey;

  // Connect to the Blob service.
  blobService = azure.createBlobService(
    accountName,accountKey,accountName + '.blob.core.windows.net');

  createContainer();
  createPolicies();

  var sasResponse = GetSAS();
  return request.respond(201, sasResponse);
}
```

For our purposes, the SAS the Node.JS application returns to the client is tied to the storage account we previously provisioned. As such, the Node.js application will need the storage account's account name and management key.

Azure Mobile Services lets you externalize configuration settings from your Node.js application. It's generally considered bad programming practice to hardcode account key settings directly in code. App settings let you set key-value pairs the service runtime can read. You can learn more about it at bit.ly/1pwGFRN.

> The easiest way to test the Raspberry Pi is to simply plug in the external monitor, keyboard, optional mouse and power supply.

Ultimately, the Node.js code will return a URL with the SAS the client can use to upload photos. Along with the SAS token, the Node.js app returns an HTTP status of 201. This means it has fulfilled that request and created a new resource (the SAS token).

You'll find a more detailed walk-through that addresses Node.js, SAS URL and the app settings at bit.ly/WyQird.

Inside the Raspberry Pi Code

The hardware/software interface is actually quite straightforward. You can think of the client-side code as a giant state machine. It's in a constant loop checking the status of the GPIO pins. As you can see from the code in **Figure 6**, we're just checking to see if a pin is on or off, true or false.

There are two ways to check pin status. The first approach, and the one we've taken here, is to talk to the file system. By checking for specific files with specific values, we can determine if a pin is on or off. Another approach, which typically offers higher performance, is to check specific locations in memory for specific values. Although it's slower to work with the file system, it's considered safer. This approach leverages the built-in driver Linux provides.

Let's address some of the code that will run on top of Mono on the Raspberry Pi using C#. The Raspberry Pi will perform four basic steps. First, it will respond to someone pressing the doorbell button. Second, based on the doorbell button being pressed, it will take a photo and save it to local storage. Third, it will request an SAS token it will then use to upload a photo. Fourth, it will upload the photo to Azure Storage as a blob. Although the code is written in C#, it's all based on REST. This means any language or environment capable of HTTP can perform these four steps.

We can control the Raspberry Pi by communicating with the GPIO. There are only four points at which our application is going to use GPIO 2 for input and GPIO 1 for output. We'll present some C# code to interface with these pins. Take a look at the main loop of the Raspberry Pi C# code in **Figure 6**. Pins 17 and 22 are used to test for device readiness and to take and upload a photo based on the doorbell button being pressed.

This infinite long-polling loop is constantly checking pin voltages. So the software knows something needs to be accomplished—like taking a photo. This loop runs at around 10hz as it polls the pin status. If you're interested in learning more about benchmarking speed, go to bit.ly/1trFzt9.

Various languages operate at different speeds. For example, Python is fairly slow and you can't use it in scenarios where speed is crucial. In this case, 10hz is fast enough to check if the doorbell switch has pulled pin 22 to Power when someone presses it.

Each of the three pins performs a different function. The first input, pin 17, must be connected to the Ground pin (which is interpreted by C# as false). If this isn't configured this way, the program will exit. Pin 22 must be set to Power (which is interpreted as true). When pin 22 is true, the photo-taking process starts in TakeAndSendPicture. The showReady method uses pin 4, an output pin, to show the program is ready to shoot and upload pictures by driving an LED.

Figure 6 **Long-Polling Loop Interfacing with the Raspberry Pi GPIO to Control Behavior**

```
while (true)
{
  // If pin 17 is off, we are ready.
  if (!s_Gpio.InputPin(FileGPIO.FileGPIO.enumPIN.gpio17))
  {
    showReady(true);
    // If pin 22 is true, indicate to the user that the system is busy,
    // take a photograph and then upload to the cloud.
    if (s_Gpio.InputPin(FileGPIO.FileGPIO.enumPIN.gpio22))
    {
      showReady(false);
      TakeAndSendPicture();
    }
  }
  else
  {
    // Not ready to take a photo yet.
    showReady(false);
    break;
  }

  // Pause for one-tenth of a second.
  System.Threading.Thread.Sleep(100);
}
```

Desktop. Web. Mobile. Your next great app starts here.

From interactive Desktop applications, to immersive Web and Mobile solutions, development tools built to meet your needs today and ensure your continued success tomorrow.

Download your free 30-day trial today and Experience the DevExpress Difference.

www.devexpress.com/try

DevExpress®

WIN ASP WPF SL VCL XAF CR

The solution leverages conditional compilation symbols so the appropriate code runs on the Raspbian OS, versus the normal Windows/Desktop OS. The code in the solution has to be different because taking a picture under Raspbian OS is not an API call. It involves running an executable in its own process. Under Windows 8, on the other hand, you leverage the CaptureFileAsync API of CameraCaptureUI.

To take the photo, we call into the built-in raspistill executable as shown in **Figure 7**. This is already configured to snap a photo and save it as a JPEG. This lets you specify the filename, photo quality and the image size. Raspistill is started as a child process. While the photo is being taken, the code requests an SAS URL from the Azure Mobile Service. It then uploads the picture to Azure Blob Storage.

The Visual Studio solution is built to run on a Raspberry Pi, as well as on a traditional Windows OS. The solution leverages conditional compilation symbols so that the appropriate code runs on the Raspbian OS versus the normal Windows desktop OS. The code in the solution has to be different because taking a picture under Raspbian OS is platform-specific. To keep things simple, when compiled for Windows 8, we don't take a photo. Instead, we upload an existing image (testPhoto.jpg) to the cloud.

Figure 7 **Taking a Photo and Uploading It to the Cloud**

```
static void TakeAndSendPicture()
{

#if LINUX
  // Start photo-taking process. This will kick off the raspistill process.
  // We'll wait for it to exit after we get the photo URL.
  Process raspistill = new Process();
  raspistill.StartInfo = new ProcessStartInfo("/usr/bin/raspistill",
    "-n -q " + photoQuality +
    " -o /home/pi/Desktop/me.jpg -h 200 -w 200 -t 500")
    {
      UseShellExecute = false
    };

    raspistill.Start();
#endif

  // Get Photo URL while the picture is being taken.
  WebRequest photoRequest = WebRequest.Create(
    "https://raspberrypiservice.azure-mobile.net/api/getuploadblobsas?");
  photoRequest.Method = "GET";
  photoRequest.Headers.Add("X-ZUMO-APPLICATION", "AIzNtpTdQLjORKJJhTrQWWRSHSnXcN78");
  PhotoResponse photoResp = null;
  using (var sbPhotoResponseStream = photoRequest.GetResponse().GetResponseStream())
  {
    StreamReader sr = new StreamReader(sbPhotoResponseStream);

    string data = sr.ReadToEnd();

    photoResp = JsonConvert.DeserializeObject<PhotoResponse>(data);
  }

  Console.WriteLine("Pushing photo to SAS Url: " + photoResp.sasUrl);
  WebRequest putPhotoRequest = WebRequest.Create(photoResp.sasUrl);
  putPhotoRequest.Method = "PUT";
  putPhotoRequest.Headers.Add("x-ms-blob-type", "BlockBlob");

#if LINUX
  // Wait until the photo is taken.
  raspistill.WaitForExit();
  FileStream fs = new FileStream(@"/home/pi/Desktop/me.jpg", FileMode.Open);
#else
  FileStream fs = new FileStream(@"testPhoto.jpg", FileMode.Open);
#endif

  using (fs)
  using (var reqStream = putPhotoRequest.GetRequestStream())
  {
    Console.WriteLine("Writing photo to blob...");
    fs.CopyTo(reqStream);
  }
  using (putPhotoRequest.GetResponse())
  {

  }
}
```

> Less code in a second language is better than lots of code in the same language.

The good news is you can share a lot of code between Windows and Raspbian. Everything from acquiring the SAS to uploading the photo is identical. So if it works in Windows, it will most likely work the same way on Raspbian. This dramatically simplifies client-based code development and testing. The Raspberry Pi is a constrained device, so it's best to minimize the work done on the device itself.

Once the device takes a photo, you should be able to see it in two places. The first place is on the device itself, specifically in /home/pi/Desktop/me.jpg, where it was specified on the command line. Then, of course, the whole point is to upload the image to the Azure Storage Account container.

Wrapping Up

This is a solid starting point for building your own Internet of Things device. We've tried to address everything from procuring the hardware, assembly, installing software, writing software and testing functionality. We've also supplied many supporting blog posts. But this solution isn't complete.

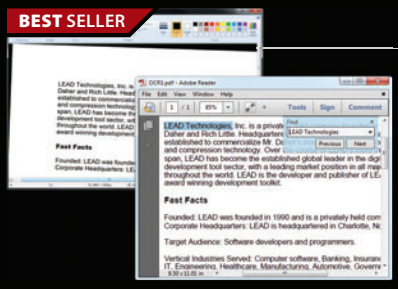
In the next article, we'll address the way push notifications are sent to phones so you can view the photo of the person ringing the doorbell on a mobile device. We'll introduce the notification service, which is part of the Azure Service Bus. We'll also build a Windows Phone application capable of receiving push notifications from Azure. Finally, we'll address additional storage mechanisms, such as using a MongoDB NoSQL database and interfacing with a face recognition back end. Stay tuned for more Internet of Things with Azure. ■

Bruno Terkaly *is a developer evangelist for Microsoft. His depth of knowledge comes from years of experience in the field, writing code using a multitude of platforms, languages, frameworks, SDKs, libraries and APIs. He spends time writing code, blogging and giving live presentations on building cloud-based applications, specifically using the Microsoft Azure platform. You can read his blog at blogs.msdn.com/b/brunoterkaly.*

Steven Edouard *is a developer evangelist for Microsoft. Before that, he worked on the .NET runtime team as a software test engineer delivering products like the .NET Framework 4.5 and .NET Native Compilation. Now his passion resides in exciting people on the limitless potentials of cloud computing services through engaging technical demonstrations, online content and presentations.*

Thanks to the following Microsoft technical experts for their review: *Gil Isaacs and Brent Stineman*

BEST SELLER



LEADTOOLS Document Imaging SDKs V18 | from $2,695.50

LEAD TECHNOLOGIES

Add powerful document imaging functionality to desktop, tablet, mobile & web applications.

- Comprehensive document image cleanup, preprocessing, viewer controls and annotations
- Fast and accurate OCR, OMR, ICR and Forms Recognition with multi-threading support
- PDF & PDF/A Read / Write / Extract / View / Edit
- Barcode Detect / Read / Write for UPC, EAN, Code 128, Data Matrix, QR Code, PDF417
- Zero-Footprint HTML5/JavaScript Controls & Native WinRT Libraries for Windows Store

BEST SELLER

ComponentOne Studio Enterprise 2014 v2 | from $1,315.60

ComponentOne

.NET Tools for the Professional Developer: Windows, HTML5/Web, and XAML.

- Hundreds of UI controls for all .NET platforms including grids, charts, reports and schedulers
- Includes 40+ UI widgets built with HTML5, jQuery, CSS3 and SVG
- New Sparkline control for HTML5 and Web Forms and options for exporting Data views
- New tools to connect, manage and display data in modern Web & Windows interfaces
- All Microsoft platforms supported, Visual Studio 2013, ASP.NET, WinForms, WPF & more

BEST SELLER

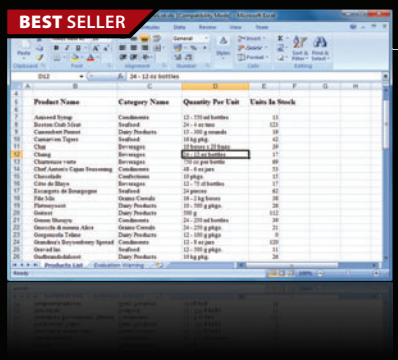
Help & Manual Professional | from $583.10

ec software

Easily create documentation for Windows, the Web and iPad.

- Powerful features in an easy accessible and intuitive user interface
- As easy to use as a word processor, but with all the power of a true WYSIWYG XML editor
- Single source, multi-channel publishing with conditional and customized output features
- Output to HTML, WebHelp, CHM, PDF, ePUB, RTF, e-book or print
- Styles and Templates give you full design control

BEST SELLER



Aspose.Total for .NET | from $2,449.02

ASPOSE
Your File Format APIs

Every Aspose .NET component in one package.

- Programmatically manage popular file formats including Word, Excel, PowerPoint and PDF
- Work with charts, diagrams, images, project plans, emails, barcodes, OCR and OneNote files alongside many more document management features in .NET applications
- Common uses also include mail merging, adding barcodes to documents, building dynamic reports on the fly and extracting text from PDF files

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

LIVE! 360

TECH EVENTS WITH PERSPECTIVE

ORLANDO
ROYAL PACIFIC RESORT AT
UNIVERSAL ORLANDO

NOV 17-21

IT AND DEVELOPER TRAINING THAT'S OUT OF THIS WORLD.

Live! 360 brings together five conferences, and the brightest minds in IT and Dev, to explore leading edge technologies and conquer current ones. These co-located events will incorporate knowledge transfer and networking, along with out-of-this-world education and training, as you create your own custom conference, mixing and matching sessions and workshops to best suit your needs.

Look for the FULL Live! 360 Pull-Out Agenda on Page 33

EVENT PARTERS Magenic Microsoft

PLATINUM SPONSOR esri

GOLD SPONSORS New Relic. sitecore Own the experience

SUPPORTED BY SharePoint Microsoft SQL Server

5 GREAT CONFERENCES. 1 GREAT PRICE.

Visual Studio LIVE!

EXPERT SOLUTIONS FOR .NET DEVELOPERS

Calling all .NET Developers: The Code is Strong with This One; Experience unbiased and practical training on the Microsoft Platform.

SharePoint LIVE!

TRAINING FOR COLLABORATION

USS Collaborate: Let's Work Together. Climb aboard to work through your most pressing SharePoint projects.

SQL Server LIVE!

TRAINING FOR DBAs AND IT PROS

SQL Server Live! will leave you with the skills needed to Conquer Planet Data, whether you are a DBA, developer, IT Pro, or Analyst.

Modern Apps LIVE!

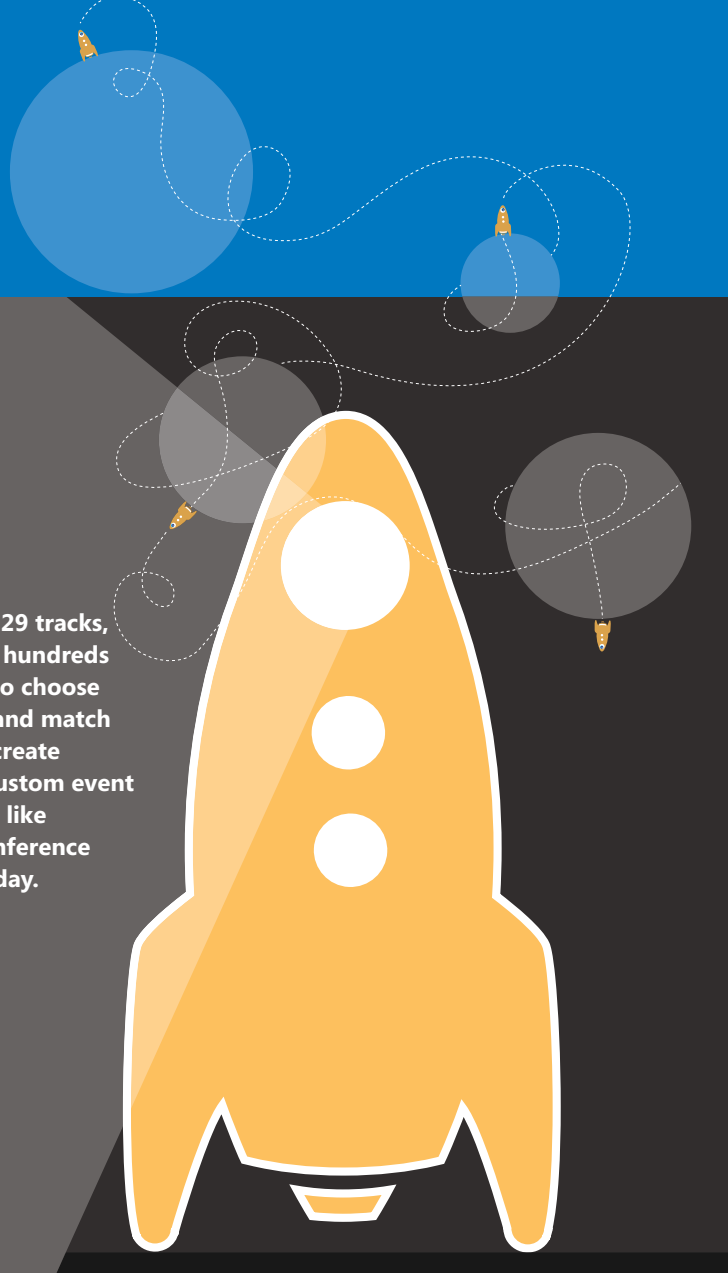
MODERN APPS FROM START TO FINISH

Master the Modern Apps Landscape and learn how to architect, design and build a complete Modern Application from start to finish.

TECHMENTOR

IN-DEPTH TRAINING FOR IT PROS

Join us for an IT Training Odyssey, focused entirely on making your datacenter more modern, more capable, and more manageable.



Five events, 29 tracks, and literally hundreds of sessions to choose from – mix and match sessions to create your own, custom event line-up – it's like no other conference available today.

REGISTER BY SEPTEMBER 17 AND SAVE $500!

Use promo code L360SEP2

Scan the QR code to register or for more event details.

CONNECT WITH LIVE!360

twitter.com/@live360events

facebook.com – Search "Live 360"

linkedin.com – Join the "Live 360" group!

msdn magazine | Visual Studio | Visual Studio MAGAZINE

PRODUCED BY 1105 MEDIA

live360events.com

Introduction to Machine Learning Studio

James McCaffrey

There's no unanimous agreement on exactly what the term "machine learning" (ML) means. In my mind, ML is any system that uses data to help make predictions. For example, you might want to predict who will win the Super Bowl, or to which group (cluster) of people a new customer will be most similar.

Writing ML systems from scratch using C# or any other programming language is fascinating, but it's time-consuming, requires specialized knowledge and is often difficult. The new Microsoft Azure ML Studio (released in July 2014) makes creating ML systems much easier, faster and more efficient. In this article, I'll walk you through a complete example that will get you up and running with ML Studio.

This article discusses:

- What the Machine Learning Studio is all about
- Setting up the data
- Creating the experiment
- Training and evaluating the model
- Making predictions

Technologies discussed:

Machine Learning Studio

Code download available at:

msdn.microsoft.com/magazine/msdnmag0914

The best way to see where this article is headed is to examine the screenshot in **Figure 1**. The image shows a completed ML Studio experiment. The goal of the experiment is to predict the political party affiliation (Democrat or Republican) of a member of the U.S. House of Representatives, based on previous voting behavior.

At the top of the image, notice that ML Studio is running in Internet Explorer—it's a Web-based application. More specifically, ML Studio is the front end for the Microsoft Azure Machine Learning service. From here on, for simplicity, I'll use the term "ML Studio" to refer to both the client front end and the Azure back end. In the address bar, you can see that I'm using an internal URL "passau.cloupapp.net." During development, the ML Studio project was code-named "Passau" and you might come across that term in the documentation. By the time you read this article, the public URL for ML Studio will be available at azure.microsoft.com.

Using ML Studio to create ML systems is roughly analogous to using Visual Studio to create executable programs, though you shouldn't get too carried away with this notion. To use Visual Studio, you can either buy the tool or use a free trial version. With ML Studio, you're charged for using the service, but there will be ways to try the system out for free. The exact details are certain to change frequently—constant change is one of the major downsides, in my opinion, of working with cloud-based systems. I like to install a product on my desktop and have any changes be totally my decision. In the brave new world of cloud computing, you have to be prepared for a working environment where change is no longer completely under your control.

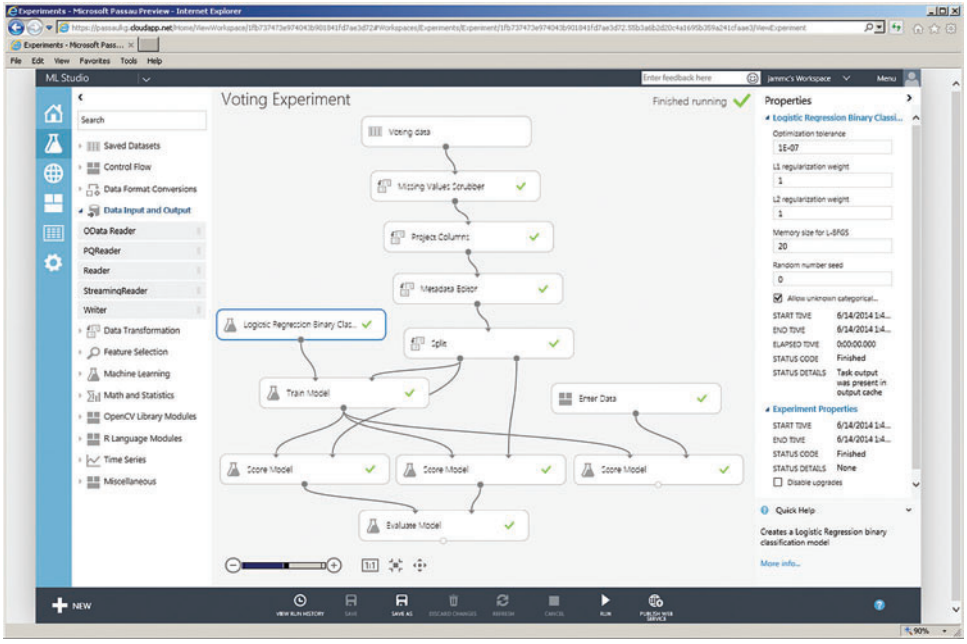


Figure 1 **A Complete Azure ML Studio Experiment**

ML Studio has three primary working areas. On the left you can see items with names like Saved Datasets, Data Input and Output, and Machine Learning. These are categories and if you expand them, you see specific items that can be dragged onto the center design surface. This is somewhat similar to the Visual Studio Toolbox, where you can drag UI controls onto a design surface. However, ML Studio modules typically represent what you can think of as methods—that is, prewritten code that performs some sort of ML task.

The center area of ML Studio is called the experiment. This is analogous to the Visual Studio editor—the place where you do most of your work. In **Figure 1**, the experiment is titled Voting Experiment. An experiment title is roughly analogous to a Visual Studio Solution name. The rectangular boxes are modules that were dragged onto the design surface. For example, the module labeled "Voting data" is the raw data source, and the module labeled "Logistic Regression Binary Classification Model" (the label is partially cut off) is the core ML algorithm used.

The curved lines establish input-output flows between modules. To be honest, my first impression as a developer was not altogether positive: "Oh great. Curvy lines. I don't like curvy lines. That's not real programming." But it didn't take me long to adapt to the ML Studio visual style of creating systems, and now I am a Believer.

The right-hand side of ML Studio shows details about whatever is currently selected in the main work area. In **Figure 1**, because the Logistic Regression Binary Classification Model module is selected (its border is bolded), the information in the right-hand area, such as "Optimization tolerance," with value 1.0E-07, refers specifically to that module. You can think of the information in the right-hand area as the parameter values (or equivalently, argument values, depending on your point of view) of the selected module/method.

You can run an experiment by clicking on the Run icon located at the bottom of the tool. This is somewhat equivalent to hitting the F5 key in Visual Studio to execute a program in the debugger. As each module finishes, ML Studio displays a green checkmark inside the module. You can also see a Save icon but, by default, ML Studio automatically saves your experiment every few seconds—working in the cloud can be hazardous due to issues like dropping a network connection.

In the sections that follow, I'll walk you through the creation of the experiment in **Figure 1** so you'll be able to replicate it. Doing so will give you a solid basis for investigating ML Studio on your own, or for exploring the early-release documentation. This article assumes you have at least beginning-level programming skills (in order to understand ML Studio and Visual Studio analogies and terminology), but does not assume you know anything about ML Studio or machine learning.

Getting the Results?

If you're new to ML Studio, you're probably wondering where to find the output of the experiment. As it turns out, a typical ML Studio experiment often has multiple outputs. The bottom-line output, so to speak, is shown in **Figure 2**. I chopped out the center section to make the image a bit easier to view.

	political-party	handicapped-infants	water-project	adopt-budget	physician-fees	el-salvador	duty-free	south-africa	Scored Labels	Scored Probabilities
Mean										0.3521
Median										0.3521
Min										0.0013
Max										0.7029
Standard Deviation										0.4961
Unique Values	1	1	2	1	2	1	2	1	2	2
Missing Values	0	0	0	0	0	0	0	0	0	0
Feature Type	String Label	String	String	String	String	String	String	String	String	Numeric
	unknown-party	y	n	y	n	y	n	y	n	democrat	0.00131325
	unknown-party	y	y	y	y	y	y	n	n	republican	0.702881

Figure 2 **Azure ML Studio Experiment Results**

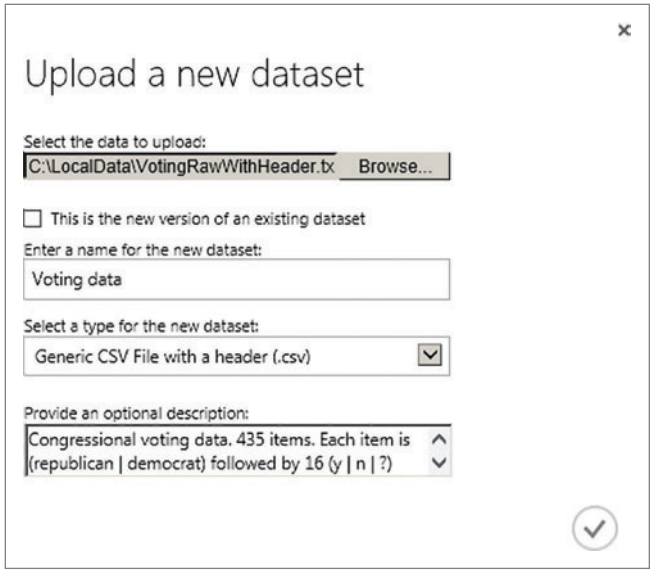


Figure 3 **Creating a New Dataset**

In order to see these results, I right-clicked on the right-most Score Model experiment module and selected the Visualize option from the context menu. This opened a separate window with the results as shown. For now, look at the bottom part of the image, which resembles:

```
unknown-party  y  n  y  n . . y  n  democrat    0.0013
unknown-party  y  y  y  y . . n  n  republican  0.7028
```

> It's fairly safe to say that all ML Studio experiments start with some data, and one or more questions to be answered.

This output indicates that after the prediction model was created, it was presented with two new data items. The first, with an unknown party, is data for a hypothetical Representative who voted "yes" on a legislative bill related to handicapped infants (the columns have headers if you look closely), "no" on a bill related to a water project, and so on, through a "no" vote on a bill related to South Africa. The model created by ML Studio predicts the hypothetical Representative is a Democrat. The second data item is for a hypothetical Representative who voted "yes" on the first eight bills and "no" on the second eight bills; the model predicts the person is a Republican.

Setting up the Data

Now that you understand the goal of the demo experiment, you're in a better position to understand how to create the experiment. It's fairly safe to say that all ML Studio experiments start with some data, and one or more questions to be answered. Here, the demo data is a well-known (to the ML community, at least) benchmark data set often called the Congressional Voting Records Data Set (or the UCI Voting Data Set, because the primary location of the file is on a server maintained by the University of California, Irvine). The raw data, a simple text file named house-votes-84.data, can be found by doing an Internet search.

The first four lines of the raw data are:

```
republican,n,y,n,y,y,y,n,n,n,y,?,y,y,y,n,y
republican,n,y,n,y,y,y,n,n,n,n,n,y,y,y,n,?
democrat,?,y,y,?,y,y,n,n,n,n,y,n,y,y,n,n
democrat,n,y,y,n,?,y,n,n,n,n,y,n,y,n,n,y
...
```

There are a total of 435 comma-delimited lines of data, one for each of the 435 members of the U.S. House of Representatives in 1984. The first column/field is the party and is either democrat or republican (there were no independents or other parties at the time). The next 16 items on each line represent a yes vote (y), a no vote (n) or a missing vote (?).

ML Studio can read data directly off the Web, or from Azure storage, but I prefer to create my own data store. To do so, I copied the text file into Notepad on my local machine, and then added column headers based on the file description on the UCI Web site, like so:

```
political-party,handicapped-infants, . . ,south-africa
republican,n,y,n,y,y,y,n,n,n,y,?,y,y,y,n,y
republican,n,y,n,y,y,y,n,n,n,n,n,y,y,y,n,?
democrat,?,y,y,?,y,y,n,n,n,n,y,n,y,y,n,n
...
```

When writing ML code from scratch, working with column headers can be annoying, so headers are often left off data files. But with ML Studio, using column headers is actually easier than omitting them, as well as making the data easier to understand. I renamed the local file to VotingRawWithHeader.txt and saved it on my machine. If you want to use the same headers as I did, you can get the data file I used in the code download for this article at msdn.microsoft.com/magazine/msdnmag0914.

After navigating to the ML Studio homepage, I clicked on the Datasets category in the left-hand pane. In the main working area, ML Studio displays a list of built-in data sets, for example Iris Two Class Data and Telescope Data. Most of these data sets you initially see are more or less well-known benchmark sets (many from the UCI repository) that can be used for exploring ML Studio. In the lower-left corner of ML Studio, I located the New icon and clicked on it.

From there I could choose either a new Dataset or a new Experiment, so I clicked on Dataset and then on the From Local File icon. This brought up the dialog box shown in **Figure 3**. I used the Browse button to target the local file, named the data set "Voting data," selected type "Generic CSV file with a header (.csv)" and typed in a brief description of the data set.

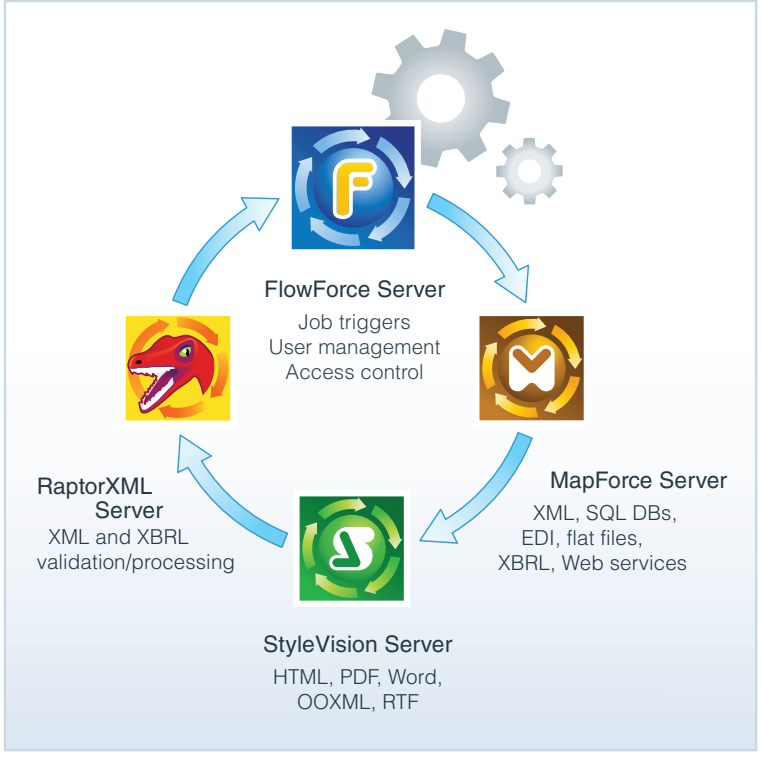
I clicked on the OK checkmark and ML Studio uploaded the local file into Azure storage and saved it. Back in the Datasets view in

Figure 4 **Processing the Data**

Manage Information Workflows with Altova® FlowForce® Server

ALTOVA®
FlowForce®
SERVER 2014

Introducing FlowForce Server, the powerful new platform for automating today's multi-step, enterprise-level data mapping, transformation, integration, and reporting tasks. This flexible workflow orchestration tool works with any combination of XML, database, flat file, EDI, Excel, XBRL, and/or Web service data common to many essential business processes.



FlowForce Server is at the center of Altova's new line of cross-platform server products optimized for today's high-performance, parallel computing environments:

- **FlowForce® Server** for orchestrating events, triggers, and the automation of processes
- **MapForce® Server** for automating any-to-any data mapping and aggregation processes
- **StyleVision® Server** for automating business report generation in HTML, PDF, and Word
- **RaptorXML® Server** for hyper-fast validation/processing of XML, XBRL, XSLT, and XQuery

Learn more and download a free trial at www.altova.com/server

ML Studio, I did a page refresh and the voting data was now visible along with the demo data sets. Note that in the pre-release version of ML Studio I used, it wasn't possible to delete a Dataset. So, when you're investigating, I strongly suggest that you create a single data set with a generic name like Dummy Data. Then, when you need a different data set, use the "This is a new version of an existing dataset" option so your ML Studio workspace doesn't become overrun with orphaned, dummy data sets that can't be deleted.

Creating the Experiment

To create the experiment, I clicked on the New icon in the lower-left corner of ML Studio, and then on the Experiment option. Next, in the left-hand pane, I clicked on the Saved Datasets category, and then scrolled to the Voting Data item I just created and dragged it onto the design pane. At the top of the design surface, I entered Voting Experiment as the title. At this point, you could right-click on the bottom output node of the Voting Data module and select the Visualize option to verify your data set is correct.

Many developers, including me, when first working with ML, seriously underestimate how much effort is involved in manipulating the source data before applying ML algorithms. Typical tasks include rearranging data columns, deleting unwanted columns, dealing with missing values, encoding non-numeric data, and splitting data into training and test sets. From a developer's point of view for the voting-data experiment, these tasks might take the form of code like this:

```
string[][] rawData = LoadData("VotingRawWithHeader.txt");
rawData = ProcessMissing(rawData, '?', 'n');
rawData = SwapColumns(rawData, 0, 16);
double[][] data = Encode(rawData);
double[][] trainData;
double[][] testData;
MakeTrainTest(data, 0.80, out trainData, out testData);
```

Figure 4 shows a close-up of the first four ML Studio modules that perform these tasks. In many ML scenarios, the most common approach to deal with missing values is to simply delete all data item rows that contain one or more missing values, and ML Studio gives you that option. However, with voting data, my hypothesis was that a missing vote was really an implied "no" vote. So, for the Missing Values Scrubber module, in the right-hand pane, I specified that all missing values ("?") should be replaced by "n" values.

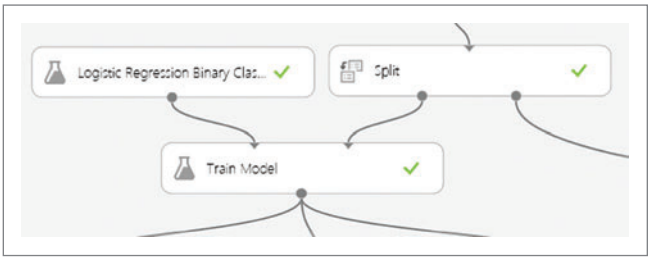


Figure 5 **Training the Model**

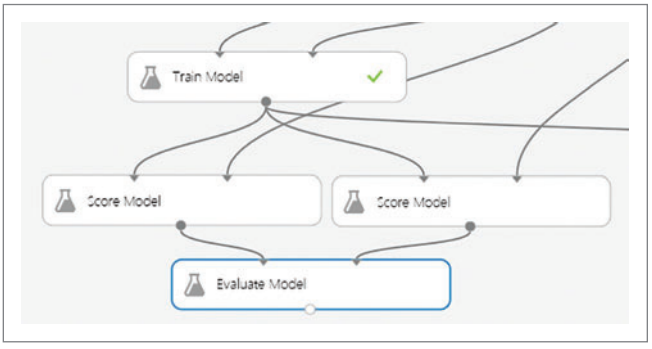


Figure 6 **Scoring and Evaluating the Model**

The Project Columns module allows you to specify any columns you want to omit. In this case, I selected the "Select all columns" option. ML Studio examines your data and makes intelligent guesses as to whether column values are string categorical data or numeric data. The Metadata Editor module allows you to override the ML assumptions and also allows you to specify the Label column, that is, the variable to predict. I selected the "political-party" column (here's where having column headers is a big help) and specified it was the Label column. I left the other 16 columns as Feature (predictor) columns.

> However, with voting data, my hypothesis was that a missing vote was really an implied "no" vote.

The Split module does just that, dividing data into a training set, used to create an ML model, and a test set, used to estimate the accuracy of the model. Here, I specified 0.8 in the module's parameter pane so the training data would be 80 percent of the 435 items (348 items) and the test set would be the remaining 20 percent (87 items). The Split module also has a Boolean parameter named "Stratified split." When working with ML Studio, you'll certainly come across parameters whose meaning you don't understand. The question-mark icon in the lower-right gives you access to the ML Studio Help.

Training the Model

You can think of an ML model as a collection of information—typically numeric values called weights—that are used to generate outputs and predictions. Training a model is the process of finding a set of weight values so that when presented with input data from the training set (in this case, 16 yes and no votes), computed outputs (either democrat or republican) closely match the known outputs in the training data. Once these weights have been determined, the resulting model can be presented with the test data. The accuracy of the model on the test data (the percentage of correct predictions) gives you a rough estimate of how well the model will do when presented with new data, where the true output isn't known.

For the voting demo, a code-based approach to training might resemble:

```
int numFeatures = 16;
LogisticModel lm = new LogisticModel(numFeatures);
int maxEpochs = 10000;
lm.Train(trainData, maxEpochs);
```

Figure 5 shows a close-up of the equivalent ML Studio training-related modules. In the demo, the Train Model module accepts as input the Logistic Regression Binary Classification Model module. Unlike the other connections, this isn't really a data flow; it actually

LIVE! 360
TECH EVENTS WITH PERSPECTIVE

ORLANDO
ROYAL PACIFIC RESORT AT
UNIVERSAL ORLANDO

NOV 17-21

IT AND DEVELOPER TRAINING THAT'S OUT OF THIS WORLD.

Live! 360 brings together five conferences, and the brightest minds in IT and Dev, to explore leading edge technologies and conquer current ones. These co-located events will incorporate knowledge transfer and networking, along with out-of-this-world education and training, as you create your own custom conference, mixing and matching sessions and workshops to best suit your needs.

VIEW ALL 180+ SESSIONS INSIDE

OPEN PAGE TO VIEW ➔

TECH EVENTS WITH PERSPECTIVE

VISUAL STUDIO LIVE! TRACKS

ASP.NET	Cloud Computing	Cross-Platform Mobile Development	Data Management	JavaScript / HTML5 Client	Visual Studio / .NET Framework	Windows Client	Windows Phone

START TIME	END TIME	Visual Studio Live! Pre-Conference: Sunday, November 16, 2014
4:00 PM	9:00 PM	
6:00 PM	9:00 PM	

START TIME	END TIME	Visual Studio Live! Pre-Conference Workshops: Monday, November 17, 2014		
8:00 AM	5:00 PM	**VSM01** - Workshop: Native Mobile App Development for iOS, Android, and Windows using C# - *Marcel de Vries & Rockford Lhotka*	**VSM02** - Workshop: AngularJS in0 to 60 - *John Papa*	**VSM03** - Workshop: FromCode to Release: DevOps forDevelopers - *Brian Randell*
5:00 PM	6:00 PM			
6:00 PM	7:00 PM			

START TIME	END TIME	Visual Studio Live! Day 1: Tuesday, November 18, 2014			
8:00 AM	9:00 AM	Visual Studio Live! Keynote: To Be Announced			
9:00 AM	9:30 AM				
9:30 AM	10:45 AM	**VST01** - Getting Started with Xamarin - *Walt Ritscher*	**VST02** - Great User Experiences with CSS 3 - *Robert Boedigheimer*	**VST03** - What's New in MVC 5 - *Miguel Castro*	**VST04** - New IDE and Editor Features in Visual Studio 2013 - *Deborah Kurata*
11:00 AM	12:15 PM	**VST05** - Building Your First Universal Application for Windows Phone and Windows Store - *Brian Peek*	**VST06** - HTML5 and Modernizr for Better Web Sites - *Robert Boedigheimer*	**VST07** - What's New in Web API 2 - *Miguel Castro*	**VST08** - Visual Studio Data Tools for Developers - *Deborah Kurata*
12:15 PM	2:00 PM				
2:00 PM	3:15 PM	**VST09** - Introduction to Developing Mobile Apps for the Web Developer - *Ben Hoelting*	**VST10** - Build an Angular and Bootstrap Web Application in Visual Studio from the Ground Up - *Deborah Kurata*	**VST11** - ASP.NET MVC for Mobile Devices - *Rob Daigneau*	**VST12** - The Road to Continuous Delivery, Automated UI Testing for Web, WPF and Windows Forms - *Marcel de Vries*
3:15 PM	4:15 PM				
4:15 PM	5:30 PM	**VST13** - Creating Responsive Cross-Platform Native/Web Apps with JavaScript and Bootstrap - *Ben Dewey*	**VST14** - Hate JavaScript? TryTypeScript - *Ben Hoelting*	**VST15** - What's New in WPF 4.5 - *Walt Ritscher*	**VST16** - Understanding Dependency Injection & Writing Testable Software - *Miguel Castro*
5:30 PM	7:30 PM				

START TIME	END TIME	Visual Studio Live! Day 2: Wednesday, November 19, 2014			
8:00 AM	9:00 AM				
9:15 AM	10:30 AM	**VSW01** - Build Cross-Platform Apps with Shared Projects, CSLA .NET, and C# - *Rockford Lhotka*	**VSW02** - AngularJS Jump-Start - *John Papa*	**VSW03** - Best Practices for Self Hosting Web API Based Services - *Rob Daigneau*	**VSW04** - The Busy Developers Guide to Virtualization with Hyper-V - *Brian Randell*
10:30 AM	11:00 AM				
11:00 AM	12:15 PM	**VSW05** - Building Cordova Apps in Visual Studio 2013 - *Nick Landry*	**VSW06** - AngularJS Anywhere with Node.js - *John Papa*	**VSW07** - Slice Development Time With ASP.NET MVC, Visual Studio, and Razor - *Philip Japikse*	**VSW08** - Build It and Ship It with TFS and Release Management - *Brian Randell*
12:15 PM	1:45 PM				
1:45 PM	3:00 PM	**VSW09** - What's New in Azure for Developers - *Vishwas Lele*	VSW10 - I Just Met You, and "This" is Crazy, But Here's My NaN, So Call(me) Maybe? - *Rachel Appel*	**VSW11** - Automated Cross Browser Testing of Your Web Applications with Visual Studio CodedUI - *Marcel de Vries*	**VSW12** - Readable Code - *John Papa*
3:00 PM	4:00 PM				
4:00 PM	5:15 PM	**VSW13** - API Management - *Vishwas Lele*	**VSW14** - Creating HTML5 and Angular Websites Using Visual Studio LightSwitch - *Michael Washington*	**VSW15** - Build Real-Time Websites and Apps with SignalR - *Rachel Appel*	**VSW16** - Visual Studio Online: An Update Every Three Weeks - *Brian Randell*
8:00 PM	10:00 PM				

START TIME	END TIME	Visual Studio Live! Day 3: Thursday, November 20, 2014			
8:00 AM	9:15 AM	**VSH01** - Creating Visual Studio Cloud Business Applications for Office 365 / SharePoint 2013 - *Michael Washington*	**VSH02** - To Be Announced	**VSH03** - Games Development with Unity and Other Frameworks for Windows and Windows Phone - *Brian Peek*	**VSH04** - Managing the .NET Compiler - *Jason Bock*
9:30 AM	10:45 AM	**VSH05** - Microsoft Azure Web Sites for the Web Developer - *Eric D. Boyd*	**VSH06** - Building Rich Data Input Windows 8 Applications - *Brian Noyes*	**VSH07** - Build Your First Mobile App in 1 Hour with Microsoft App Studio - *Nick Landry*	**VSH08** - Writing Asynchronous Code Using .NET 4.5 and C# 5.0 - *Brian Peek*
11:00 AM	12:15 PM	**VSH09** - Moving Web Apps to the Cloud - *Eric D. Boyd*	**VSH10** - XAML Antipatterns - *Ben Dewey*	**VSH11** - To Be Announced	**VSH12** - Asynchronous Debugging in .NET - *Jason Bock*
12:15 PM	1:30 PM				
1:30 PM	2:45 PM	**VSH13** - Node.js for .NET Developers - Jason Bock	**VSH14** - Building Maintainable and Extensible MVVM WPF Apps with Prism 5 - *Brian Noyes*	**VSH15** - Learning Entity Framework 6 - *Leonard Lobel*	**VSH16** - Rock Your Code Using Code Contracts - *David McCarter*
3:00 PM	4:15 PM	**VSH17** - Building Mobile Cross-Platform Apps in C# with Azure Mobile Services - *Nick Landry*	**VSH18** - XAML for the WinForms Developer - *Philip Japikse*	**VSH19** - Database Development with SQL Server Data Tools - *Leonard Lobel*	**VSH20** - Rock Your Code With Visual Studio Add-ins - *David McCarter*
4:30 PM	5:45 PM				

START TIME	END TIME	Visual Studio Live! Post-Conference Workshops: Friday, November 21, 2014	
8:00 AM	5:00 PM	**VSF01** - Workshop: Service Orientation Technologies: Designing, Developing, & Implementing WCF and the Web API - *Miguel Castro*	**VSF02** - Workshop: Build a Windows 8.1 Application in a Day - *Philip Japikse*

Speakers and sessions subject to change

SharePoint LIVE!
TRAINING FOR COLLABORATION

SQL Server LIVE!
TRAINING FOR DBAs AND IT PROS

SHAREPOINT LIVE! TRACKS

Developing Apps and Solutions for SharePoint	High-Value SharePoint Workloads: Social, Search, BI, and Business Process Automation	Information and Content Management: Documents, Records, and Web	Keys to SharePoint Success: Strategy, Governance, Adoption	SharePoint Infrastructure Management and Administration	SharePoint, Office 365 and the Cloud

SQL SERVER LIVE! TRACKS

BI, Big Data, Data Analytics, and Data Visualization	SQL Server Administration & Maintenance	SQL Server Features & Components	SQL Server for the Developer / SQL Server 2014	SQL Server in the Cloud	SQL Server Performance Tuning and Optimization	SQL Server Tools, Tricks, and Techniques

SharePoint Live! Pre-Conference: Sunday, November 16, 2014

SQL Server Live! Pre-Conference: Sunday, November 16, 2014

Pre-Conference Registration - Royal Pacific Resort Conference Center

Dine-A-Round Dinner @ Universal CityWalk

SharePoint Live! Pre-Conference Workshops: Monday, November 17, 2014

SPM01 - Workshop: Modern Office 365, SharePoint & Cloud Development Ramp-Up - *Andrew Connell*

SPM02 - Workshop: Getting Up and Running with Office 365 - *Dan Usher*

SQL Server Live! Pre-Conference Workshops: Monday, November 17, 2014

SQM01 - Workshop: Big Data and NoSQL for Database and BI Pros - *Andrew Brust*

SQM02 - Workshop: Maintaining SQL Server for Non-DBAs - *Don Jones*

EXPO Preview

Live! 360 Keynote:

SharePoint Live! Day 1: Tuesday, November 18, 2014

SharePoint Live! Keynote: Creating Interactive Experiences for Internal and External Users in Office 365 - *Sonya Koptyev, Senior Product Manager, SharePoint and Office, Microsoft*

SQL Server Live! Day 1: Tuesday, November 18, 2014

SQL Server Live! Keynote: To Be Announced

Networking Break • Visit the EXPO

SPT01 - Lean-Agile Development with SharePoint - *Bill Ayres*	**SPT02** - What's New for IT PRO's in SharePoint 2013 - *Brian Alderman*	**SPT03** - Getting Started with Cross Site Publishing in SharePoint 2013 - *Prashant Bhoyar*	**SQT01** - Best Practices for Leveraging SQL Server in Windows Azure Virtual Machines - *Scott Klein*	**SQT02** - SQL Server Integration Services End-to-End - *David Dye*	**SQT03** - Introduction to SQL Server Clustering - *Chad Churchwell*
SPT04 - How to Improve the SharePoint UI Using Bootstrap 3 - *Ryan McIntyre*	**SPT05** - Microsoft To Be Announced	**SPT06** - To Be Announced	**SQT04** - SQL Server 2014 In-memory OLTP Deep Dive - *Scott Klein*	**SQT05** - Designing and Maintaining a SQL Server Replication Topology - *Chad Churchwell*	**SQT06** - Hadoop Administration for SQL Server DBAs - *Edwin Sarmiento*

Lunch • Visit the EXPO

SPT07 - Building SharePoint Single Page Apps with AngularJS - *Andrew Connell*	**SPT08** - Getting Started with Office 365 - Identity, Provisioning and Basics of Tenant Administration - *Dan Usher*	**SPT09** - Learn About the Top 5 Advanced Search Features You Never Knew You Needed - *Paul Olenick*	**SQT07** - Big Data 101 with Azure HDInsight - *Andrew Brust*	**SQT08** - Implementing Auditing in SQL Server - *David Dye*	**SQT09** - Building Perfect SQL Servers, Every Time - *Joey D'Antoni*

Networking Break • Visit the EXPO

SPT10 - SharePoint 2013 Display Templates and Query Rules - *Matt McDermott*	**SPT11** - Office 365 Information Architecture and Governance 101 - *Ben Curry*	**SPT12** - Getting Started with SharePoint 2013 Workflows - *Prashant Bhoyar*	**SQT10** - Integrating SQL Server Analysis Services with Hadoop - *Edwin Sarmiento*	**SQT11** - Dealing with Errors in SQL Server Integration Services - *David Dye*	**SQT12** - SQL Server High Availability and Disaster Recovery - *Chad Churchwell*

Exhibitor Reception

SharePoint Live! Day 2: Wednesday, November 19, 2014

SQL Server Live! Day 2: Wednesday, November 19, 2014

Live! 360 Keynote: To Be Announced

SPW01 - Office Graph and Oslo - The Future of Discovering and Consuming Information? - *Agnes Molnar*	**SPW02** - SharePoint Tips and Tricks - Avoiding Administrative Blunders - *Dan Usher*	**SPW03** - Real World: Hybrid Office 365 Solution - *Ben Curry*	**SQW01** - Who Moved My Tuple? Columnstore Indexes in SQL Server 2014 - *Joey D'Antoni*	**SQW02** - Integrating Reporting Services with SharePoint - *Kevin Goff*	**SQW03** - Everything You Never Wanted to Know about SQL Server Indexes - *Don Jones*

Networking Break • Visit the EXPO

SPW04 - To Be Announced	**SPW05** - SharePoint Intranet Design - *Michael Doyle*	**SPW06** - Roadmap for a Successful Migration to SharePoint Online - *Paul Olenick*	**SQW04** - Getting Started with MDX - *William E. Pearson III*	**SQW05** - To Be Announced	**SQW06** - Virtualizing SQL Server in VMware, Hyper-V, or Xen: Yes or No? - *Don Jones*

Birds-of-a-Feather Lunch • Visit the EXPO

SPW07 - Deploying Provider-Hosted Apps Into Production - *Paul Schaeflein*	**SPW08** - Optimizing SQL Server 2014 for SharePoint 2013 - *Brian Alderman*	**SPW09** - Office 365: Considerations for a Hybrid Deployment - *Geoff Varosky*	**SQW07** - Getting Started with Analysis Services 2014 Tabular - *William E. Pearson III*	**SQW08** - Programming the T-SQL Enhancements in SQL Server 2012 - *Leonard Lobel*	**SQW09** - Prevent Recovery Amnesia - Forget the Backups - *Chris Bell*

Networking Break • Visit the EXPO • Expo Raffle @ 3:30 p.m.

SPW10 - Test-driven Development with SharePoint 2013 - *Bill Ayres*	**SPW11** - Bringing the Users With You - SharePoint 2013 Adoption and Engagement - *Robert Bogue*	**SPW12** - SharePoint and the Mystical OneDrive for Business - *Dan Usher*	**SQW10** - Implementing Data Warehouse Patterns with the Microsoft BI Tools - *Kevin Goff*	**SQW11** - Accelerate Database Performance through Data Compression - *Joey D'Antoni*	**SQW12** - Building a Database Deployment Pipeline - *Grant Fritchey*

Live! 360 Evening Event

SharePoint Live! Day 3: Thursday, November 20, 2014

SQL Server Live! Day 3: Thursday, November 20, 2014

SPH01- How to Develop and Debug Client-Side Code - *Mark Rackley*	**SPH02** - Managing SharePoint 2013 with System Center Operations Manager (SCOM) 2012 R2 - *Jason Kaczor*	**SPH03** - Getting a Grip on SharePoint Social - *Michael Doyle*	**SQH01** - SQL Server Reporting Services on Power BI - *Kevin Goff*	**SQH02** - Getting Started Reading Query Execution Plans - *Grant Fritchey*	**SQH03** - Database Design: Size DOES Matter - *Thomas LaRock*
SPH04 - Deep Dive into Office 365 APIs - *Jeremy Thake*	**SPH05** - No Governance As Usual - *Robert Bogue*	**SPH06** - Is "Enterprise Search" Dead??? - *Agnes Molnar*	**SQH04** - Excel, Power BI and You: An Analytics Superhub - Andrew Brust	**SQH05** - Secrets of SQL Server: Database WORST Practices - *Pinal Dave*	**SQH06** - Securing Your Database by Using Transparent Data Encryption (TDE) - *Bradley Ball*
SPH07 - Lazy Client-side Dev: Let the 3rd-party Libraries Do the Heavy Lifting - *Mark Rackley*	**SPH08** - Automating your Enterprise Application Deployments with PowerShell - *Geoff Varosky*	**SPH09** - Implementing ECM Solutions with SharePoint 2013 - *Rob Bogue*	**SQH07** - Self-Service BI Governance - *Jen Underwood*	**SQH08** - Statistics and the Query Optimizer - *Grant Fritchey*	**SQH09** - To Be Announced

Lunch on the Lanai

SPH10 - BreezeJS Makes Client-Side SharePoint 2013 REST Development a... BREEZE! - *Andrew Connell*	**SPH11** - 10 Misconceptions of Information Architecture You Should Forget - *Agnes Molnar*	**SPH12** - Developers Approach to Search Applications - *Matt McDermott*	**SQH10** - Busy Developer's Guide to R - *Ted Neward*	**SQH11** - Inside the SQL Server Query Optimizer - *Bradley Ball*	**SQH12** - Monitoring Databases in a Virtual Environment - *Thomas LaRock*
SPH13 - SharePoint Design Manager – Step by Step -*Paul Schaeflein*	**SPH14** - SharePoint Forensics for IT Professionals - *Jason Kaczor*	**SPH15** - Introducing Office Web Apps as a Tool for Developing Content Rich Applications - *Ryan McIntyre*	SQH13 - Advanced Analytics and Data Mining for Power BI - *Jen Underwood*	**SQH14** - Cardinality Estimates in SQL Server 2014 - *Thomas LaRock*	**SQH15** - Busy Developer's Guide to NoSQL - *Ted Neward*

Live! 360 Conference Wrap-up Andrew Connell, Matt McDermott, Don Jones, Greg Shields, Andrew Brust, Rockford Lhotka

SharePoint Live! Post-Conference Workshops: Friday, November 21, 2014

SPF01 - Workshop: Apps for SharePoint - The Next Level - *Paul Schaeflein*

SPF02 - Workshop: Implementing Search Based Intranet in SharePoint 2013 and Office 365 - *Agnes Molnar*

SQL Server Live! Post-Conference Workshops: Friday, November 21, 2014

SQF01 - Workshop: Performance Tuning Best Practices - Every DBA and Developer MUST Know - *Pinal Dave*

SQF02 - Workshop: SQL Server 2014 for Developers - *Leonard Lobel*

TECHMENTOR
IN-DEPTH TRAINING FOR IT PROS

Modern Apps LIVE!
MODERN APPS FROM START TO FINISH

TECHMENTOR TRACKS							MODERN APPS LIVE! TRACK
Advanced Windows Networking	Hybrid Cloud and Virtualizatin	Jack of All Trades	Security and Ethical Hacking	System Center & Microsoft Exchange	Windows PowerShell	Windows Server	Presented in Partnership with: Magenic

TechMentor Pre-Conference: Sunday, November 16, 2014	Modern Apps Live! Pre-Conf.: Sun., Nov. 16

TechMentor Pre-Conference Workshops: Monday, November 17, 2014		MAL! Pre-Conf. Workshop: Mon., Nov. 17
TMM01 - Workshop: Microsoft Office 365: The Bootcamp - *Andy Malone*	**TMM02** - Workshop: Automating Hyper-V with System Center Technologies - *Greg Shields*	**MAM01** - Workshop: Modern App Technology Overview - Android, iOS, Cloud, and Mobile Web - *Nick Landry, Kevin Ford, & Steve Hughes*

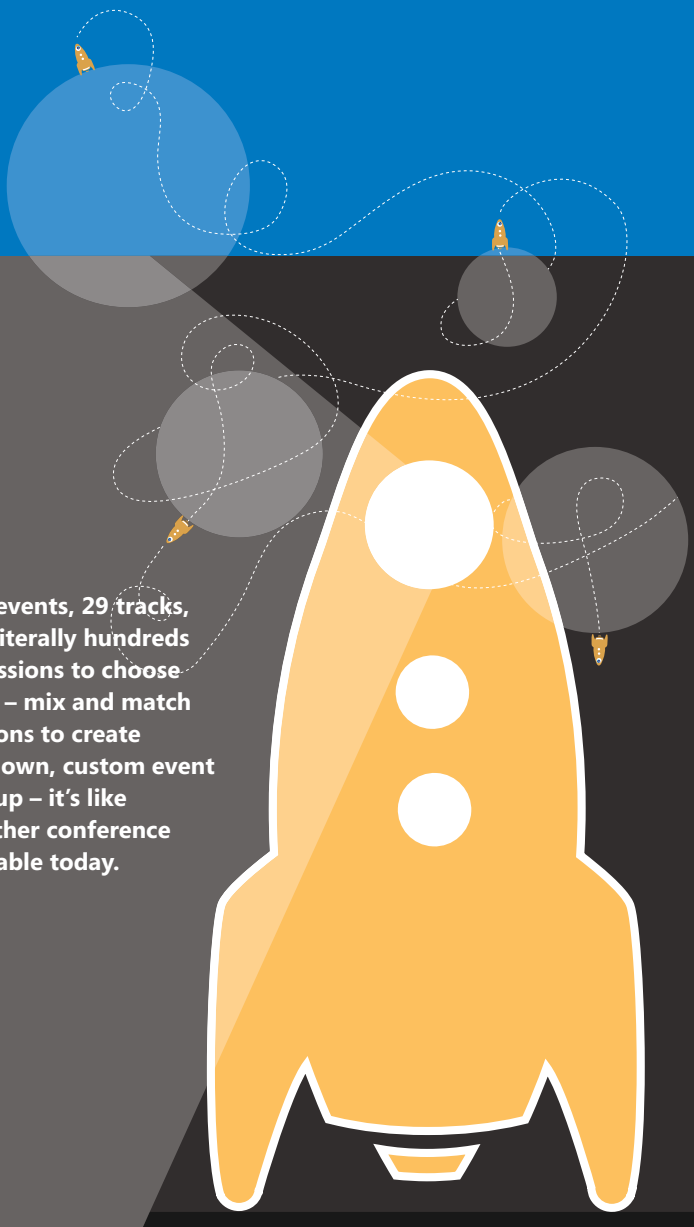
TechMentor Day 1: Tuesday, November 18, 2014			Modern Apps Live! Day 1: Tues., Nov. 18
TechMentor Keynote: To Be Announced			Modern Apps Live! Keynote: To Be Announced
TMT01 - Desired State Configuration: An Administrator's Overview - *Don Jones*	**TMT02** - Automating vSphere with VMware Orchestrator, Part 1 - *Greg Shields*	**TMT03** - Creating an Everywhere LAN Without the Hassle Using DirectAccess - *John O'Neill, Sr.*	**MAT01** - Defining Modern App Development - *Rockford Lhotka*
TMT04 - PowerShell Scripting and Toolmaking with Patterns and Practical Uses - *Don Jones*	**TMT05** - Automating vSphere with VMware Orchestrator, Part 2 - *Greg Shields*	**TMT06** - To Be Announced	**MAT02** - Modern App Architecture - *Rockford Lhotka*
TMT07 - GO DARK: The PowerShell Delegated Administration Resource Kit - *Don Jones*	**TMT08** - Using Microsoft Azure for Easy, Secure, Off-Site Backup - *John O'Neill, Sr.*	**TMT09** - System Center and the Modern Datacenter: A Tactical Briefing - *Greg Shields*	**MAT03** - ALM with Visual Studio Online (TFS) and Git - *Brian Randell*
TMT10 - Top 20 Mistakes in Microsoft Public Key Infrastructure (PKI) Deployments - *Mark B. Cooper*	**TMT11** - Integrating Office 365 with Active Directory, Step-by-Step - *John O'Neill, Sr.*	**TMT12** - Career Strategies for IT Professionals: A Roundtable Discussion and Q&A - *Don Jones & Greg Shields*	**MAT04** - Reusing Business and Data Access Logic Across Platforms - *Kevin Ford*

TechMentor Day 2: Wednesday, November 19, 2014			Modern Apps Live! Day 2: Wed., Nov. 19
TMW01 - Managing and Deploying BYOD Identity Solutions with a Microsoft PKI - *Mark B. Cooper*	**TMW02** - Managing Infrastructure Migrations at Windows Server's End-of-Life - *Rick Claus*	**TMW03** - Automating Application Installations: An Introduction to Software Packaging & Repackaging - *John O'Neill, Sr.*	**MAW01** - Coding for Quality and Maintainability - *Jason Bock*
TMW04 - Securing Cloud Servers and Services with PKI Certificates - *Mark B. Cooper*	**TMW05** - Everything You've Needed to Know about Windows Scale Out File Server - *Rick Claus*	**TMW06**- Skype Internals from a Security Perspective - *Andy Malone*	**MAW02** - UX Design for Modern Apps - *Anthony Handley*
TMW07 - CIM-ple Remote Management with PowerShell - *Jeffery Hicks*	**TMW08** - Troubleshooting the Windows Client, an Introductory Clinic - *Jeff Stokes*	**TMW09** - The Dark Side of Social Networking and How to Survive It - *Andy Malone*	**MAW03** - Applied UX: iOS, Android, Windows - *Anthony Handley*
TMW10 - Creating Graphical PowerShell Tools - *Jeffery Hicks*	**TMW11** - Building a Better VDI Image: Best Practices from Microsoft - *Jeff Stokes*	**TMW12** - The Hacker's Guide to Identity Theft and How to Survive It - *Andy Malone*	**MAW04** - Leveraging Azure Services - *Kevin Ford*

TechMentor Day 3: Thursday, November 20, 2014			Modern Apps Live! Day 3: Thurs., Nov. 20
TMH01 - Managing Hyper-V with Windows PowerShell - *Jeffery Hicks*	**TMH02** - Building a Secure and Cost-Effective BitLocker - *Sami Laiho*	**TMH03** - Improve Your Tech Writing Skills: Lessons from a Professional Author - *Don Jones*	**MAH01** - Analyzing Results with Power BI - *Steve Hughes*
TMH04 - Document Everything with Windows PowerShell - *Jeffery Hicks*	**TMH05** - The Real-World Guide to Upgrading your IT Skills AND Your Infrastructure, Part 1 - *Rick Claus*	**TMH06** - Proactive Security in Windows Environments - *Sami Laiho*	**MAH02** - Building a Native iOS App - *Lou Miranda*
TMH07 - Discussing the Future of Desktop Administration: Are Configuration Manager's Days Numbered? - *Don Jones and Greg Shields*	**TMH08** - The Real-World Guide to Upgrading your IT Skills AND Your Infrastructure, Part 2 - *Rick Claus*	**TMH09** - How to Build a Perfect File Server with Windows Server - *Sami Laiho*	**MAH03** - Building an Android App with Xamarin - *Nick Landry*
TMH10 - Eliminate the Regulatory Compliance Nightmare - *J. Peter Bruzzese*	**TMH11** - To Be Announced	**TMH12** - Oh Snap! Active Directory Attribute-level Recovery when the AD Recycle Bin Cannot Help - *Ashley McGlone*	**MAH04** - Building a Windows App - *Brent Edwards*
TMH13 - Exchange 2013 and SharePoint 2013: Better Together - *J. Peter Bruzzese*	**TMH14** - Migrating Your Windows Failover Clusters - Is Now the Right Time to Upgrade? - *Bruce Mackenzie-Low*	**TMH15** - Migrate GPOs from One Domain / Forest to another Using PowerShell - *Ashley McGlone*	**MAH05** - Building a Responsive Single Page App - *Allen Conway*

TechMentor Post-Conference Workshops: Friday, November 21, 2014		MAL! Post-Conf. Workshop: Fri., Nov. 21, 2014
TMF01 - Workshop: Microsoft Exchange Boot Camp for the Accidental Exchange Administrator - *J. Peter Bruzzese*	**TMF02** - Workshop: Windows Troubleshooting DEEP DIVE - *Bruce Mackenzie-Low*	**MAF01** - Workshop: Modern App Development In-Depth: iOS, Android, Windows, and Web - *Brent Edwards, Anthony Handley, Lou Miranda, & Allen Conway*

5 GREAT CONFERENCES. 1 GREAT PRICE.



Visual Studio LIVE!

EXPERT SOLUTIONS FOR .NET DEVELOPERS

Calling all .NET Developers: The Code is Strong with This One; Experience unbiased and practical training on the Microsoft Platform.

SharePoint LIVE!

TRAINING FOR COLLABORATION

USS Collaborate: Let's Work Together. Climb aboard to work through your most pressing SharePoint projects.

SQL Server LIVE!

TRAINING FOR DBAs AND IT PROS

SQL Server Live! will leave you with the skills needed to Conquer Planet Data, whether you are a DBA, developer, IT Pro, or Analyst.

Modern Apps LIVE!

MOBILE, CROSS-DEVICE & CLOUD DEVELOPMENT

Master the Modern Apps Landscape and learn how to architect, design and build a complete Modern Application from start to finish.

TECHMENTOR

IN-DEPTH TRAINING FOR IT PROS

Join us for an IT Training Odyssey, focused entirely on making your datacenter more modern, more capable, and more manageable.

Five events, 29 tracks, and literally hundreds of sessions to choose from – mix and match sessions to create your own, custom event line-up – it's like no other conference available today.

REGISTER BY SEPTEMBER 17 AND SAVE $400!

Use promo code L360SEPTI

Scan the QR code to register or for more event details.

CONNECT WITH LIVE! 360

twitter.com/@live360events

facebook.com – Search "Live 360"

linkedin.com – Join the "Live 360" group!

SUPPORTED BY

SharePoint | Microsoft SQL Server | msdn magazine | Visual Studio | Visual Studio MAGAZINE

LIVE! 360
TECH EVENTS WITH PERSPECTIVE

FEATURED SPEAKERS

Live! 360 brings you over 70 speakers; some of the best and brightest experts in the industry.

Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS


RACHEL APPEL

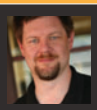

JASON BOCK


ROBERT BOEDIGHEIMER


MIGUEL CASTRO


MARCEL DE VRIES


DEBORAH KURATA


ROCKFORD LHOKTA


BRIAN NOYES


JOHN PAPA


BRIAN RANDELL

TECHMENTOR
IN-DEPTH TRAINING FOR IT PROS


JEFFERY HICKS


DON JONES


SAMI LAIHO


BRUCE MCKENZIE-LOW


GREG SHIELDS

SharePoint LIVE!
TRAINING FOR COLLABORATION


BILL AYERS

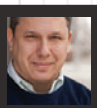

ROBERT BOGUE


ANDREW CONNELL

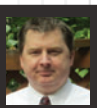

MATTHEW MCDERMOTT


AGNES MOLNAR

SQL Server LIVE!
TRAINING FOR DBAs AND IT PROS


ANDREW BRUST


PINAL DAVE


SCOTT KLEIN


LEONARD LOBEL


TED NEWARD

Modern Apps LIVE!
MOBILE, CROSS-DEVICE & CLOUD DEVELOPMENT


ALLEN CONWAY


BRENT EDWARDS


KEVIN FORD


STEVEN HUGHES


NICK LANDRY

REGISTER BY SEPTEMBER 17 AND SAVE $400!

Use promo code L360SEPTI

CONNECT WITH LIVE! 360

twitter.com/@live360events
facebook.com – Search "Live 360"
linkedin.com – Join the "Live 360" group!

VIEW THE FULL LIST OF SPEAKERS AT LIVE360EVENTS.COM

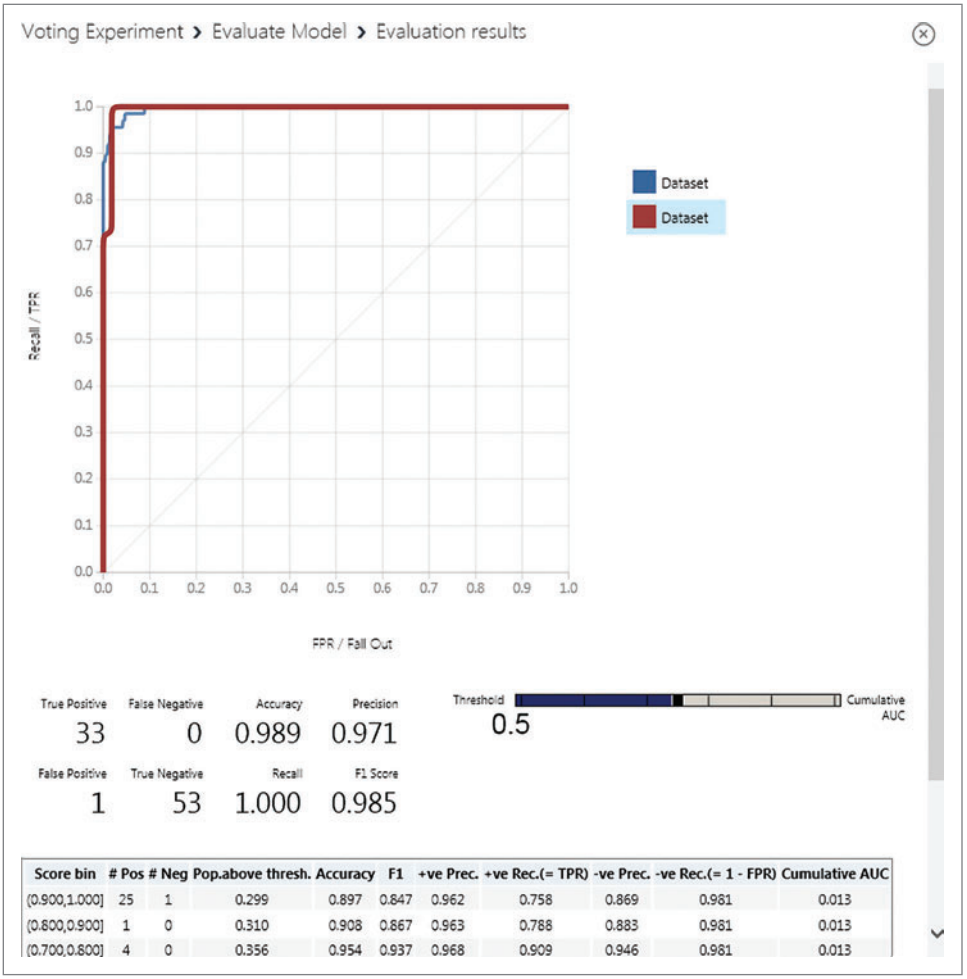


Figure 7 **Model Accuracy on Test Data**

specifies exactly what kind of ML model is to be used. Alternatives to Logistic Regression Binary Classification include modules Averaged Perceptron Binary, Boosted Decision Tree Binary and Neural Network Binary Classifiers.

So, how do you know which model to use? Because the output to predict has two possible values, democrat or republican, you want a binary model. But there are dozens of ML approaches and probably the hardest part of using ML Studio is having to research the pros and cons of different ML classifiers. A developer analogy with Visual Studio is that the Microsoft .NET Framework has dozens of data structures, such as generic Dictionary, HashSet, and generic Queue, and it's up to you to know exactly what each data structure does. In the same way, it's up to you to learn about ML classifiers.

The Logistic Regression module has some parameters you likely won't understand, including Optimization tolerance, L1 regularization weight, and Memory size for L-BFGS. Again, it's up to you to learn what these parameters do. Fortunately, ML Studio has well-chosen default values for most module parameters. I accepted all default parameter values except I used zero for the "Random number seed."

In the Train Model module, you have to tell ML Studio which column of the training data is the Label column; that is, which column is the variable to predict. With the Train Model module selected, I clicked on the Launch column selector button in the module's parameters pane and chose the pick-by-name option in the dropdown control, and then typed political party. I could've used the column index, 1, because political party is in the first column (ML Studio indices are 1-based rather than 0-based as developers are accustomed to). Note that specifying the Label column for the Train Model module is required even if you do so in the Metadata Editor.

Evaluating the Model

After the demo model has been trained, the next steps are to feed the training data and the test data to the model, calculate the computed outputs, and calculate the accuracy of the computed outputs (against the known outputs). In code, this might look like:

```
lm.ComputeOutputs(trainData); // Score
double trainAccuracy = lm.Accuracy(trainData); // Evaluate
lm.ComputeOutputs(testData); // Score
double testAccuracy = lm.Accuracy(testData); // Evaluate
```

Figure 6 shows a close-up of the relevant scoring and evaluating modules. The two Score Model modules accept two input flows. The first input is the trained model (the information needed to compute outputs), and the second input is either a training set or test set

Figure 8 **Entering New Data to Predict**

(the data inputs needed). The results of these two scoring modules are sent to the Evaluate Model module, which computes accuracy.

Figure 7 shows the results in the Evaluate Model module. Notice in the upper right, the second of two Dataset items has been selected (it's highlighted), which means the results are for the test data. The most important part of the result is the Accuracy value of 0.989. Recall the test set was 20 percent of the 435 original data items, or 87 items. The Logistic Regression model correctly predicted the political party of 86 out of the 87 test items. There's a lot of other information in Evaluate Model module results. For example, the graph is called a Receiver Operating Characteristic (ROC) graph. It plots the percentage of "true positives" (correct predictions) on the y-axis vs. the percentage of "false positives" (incorrect predictions) on the x-axis.

Making Predictions

Once an ML Studio model has been created and evaluated, it can be used to make predictions on data with unknown outputs. I wanted to predict the political party of a hypothetical Representative who voted "yes," "no," "yes," "no," and so on, on the 16 legislative bills, and a second Representative who voted "yes" on the first eight bills and "no" on the remaining eight bills.

One approach would be to create and upload a new ML Studio Dataset and then score it in the same way as the training and test data sets. But for a limited amount of data, a more interactive approach is to use the Enter Data module as shown in **Figure 8**, which allows you to enter data manually.

The format of the data in the module must exactly match the format of the data used to train the model, so the column headers are required. The output from the Enter Data module is combined with the output from the Train Model module. After running the experiment, you can see the results by clicking on the Visualize option of the Score Model module, as shown earlier in **Figure 2**.

> One of the coolest things about ML Studio is that you can write your own custom modules using C#.

If you were making predictions using a procedural programming language, the code might resemble:

```
string[] unknown = new string[] { "party", "y", "n", "y", . . "n" };
double result = lm.ComputeOutput(unknown);
if (result < 0.5)
  Console.WriteLine("Predicted party is democrat");
else
  Console.WriteLine("Predicted party is republican");
```

Again, some of the ML Studio information is likely to be a bit mysterious. Recall that the predictions include two trailing numeric values:

```
unknown-party  y  n  y . . y  n  democrat    0.0013
unknown-party  y  y  y . . n  n  republican  0.7028
```

As it turns out, for Logistic Regression Binary Classification, an output value below 0.5 indicates the first class (democrat in this example) and an output value above 0.5 indicates the second class (republican). Keep in mind that, like Visual Studio, ML Studio has many features and generates a huge amount of information. You learn what the various pieces of information mean over time by using the system and tackling one new piece at a time, rather than doing a deep dive into the documentation and trying to learn everything at once.

No Code?!

This article just scratched the surface of ML Studio, but it provides enough information so you can replicate the voting experiment and give the tool a try. You may have noticed that this article doesn't include any coding. No code? Bah! But one of the coolest things about ML Studio is that you can write your own custom modules using C#, and I'll be covering this topic in future articles. My hunch is that ML Studio will generate an ecosystem in which developers write sophisticated, specialized modules and make them available both commercially and through various channels, such as open source and blogs. ■

More Than Just a Tool

The Azure ML Studio application, together with its back-end engine, the Microsoft Azure Machine Learning (ML) service, is much more than the client tool described in this short article. From a developer's point of view, ML Studio dramatically simplifies the creation of prediction systems. But there are additional value propositions that are not so obvious.

One important topic not covered in this article is the ability of Azure ML to create and publish a Web service using just drag and drop and a few clicks. Azure ML automatically handles deployment, capacity provisioning, load balancing, auto-scaling and health monitoring. One pre-release customer estimated that with Azure ML, they were able to create a business solution (a fraud detection system) at a tiny fraction of the cost of using commercial analytics software.

Azure ML supports R, a popular data science programming language. Hundreds of existing open source R modules can be directly copied into an Azure ML system.

ML Studio allows easy collaboration. An experiment can be easily shared among several people. I've used this feature myself and it was much more efficient than my normal, back-and-forth e-mail conversation approach.

Azure ML provides more than just advantages to developers and data scientists. "Deploying advanced analytics is hard," said Joseph Sirosh, Microsoft corporate vice president, Information Management and Machine Learning. "Enterprises are tired of paying high prices, recruiting expensive talent and waiting months to get results. Having the ability to quickly develop analytic models and deploy them without these bottlenecks is game-changing. Azure ML allows businesses to unlock value in their data and build systems to reduce expenses, grow revenue and serve their end customers better."

Dr. James McCaffrey *works for Microsoft Research in Redmond, Wash. He has worked on several Microsoft products including Internet Explorer and Bing. He can be reached at jammc@microsoft.com.*

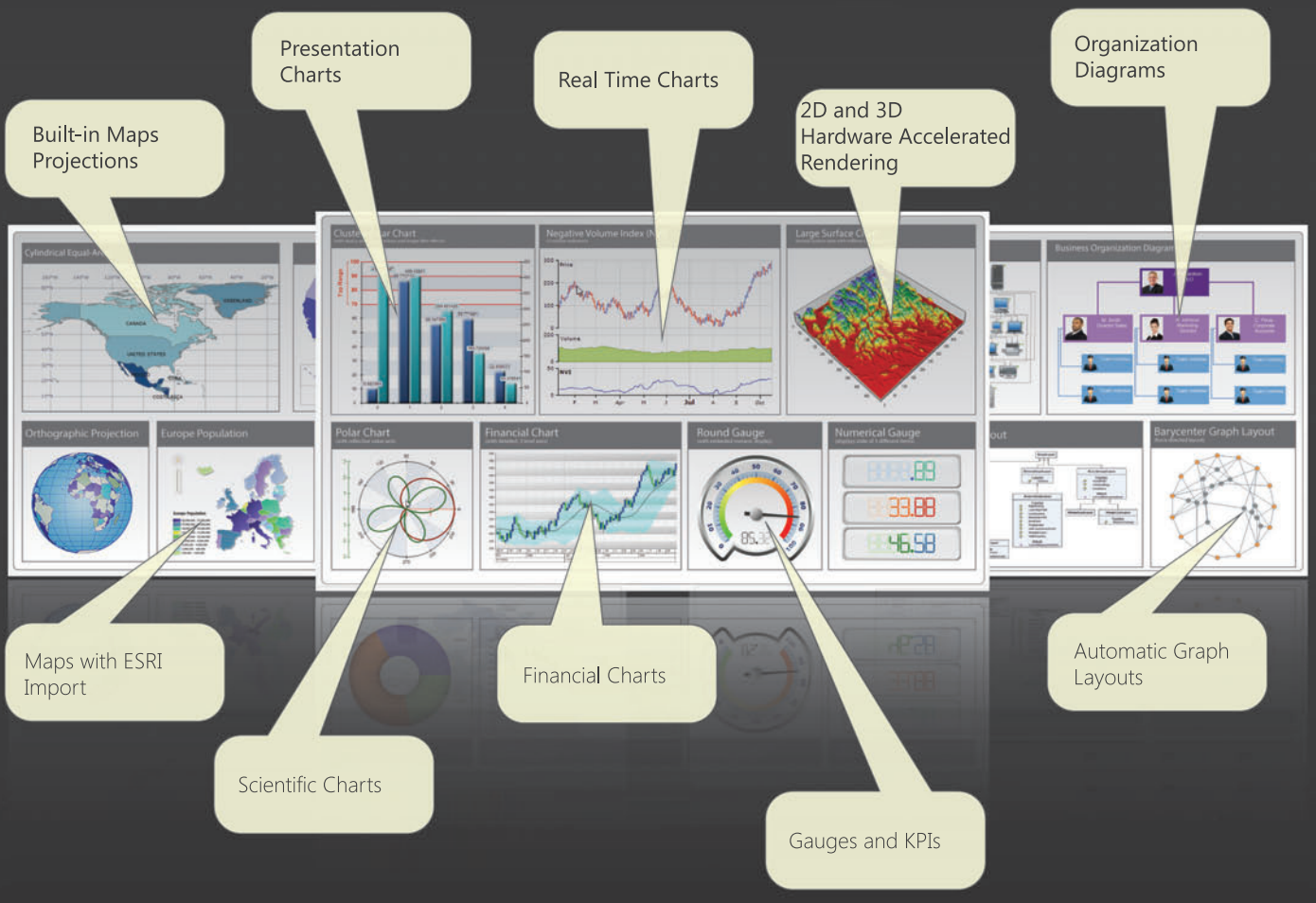
Thanks to the following Microsoft technical experts for reviewing this article: *Roger Barga (Machine Learning) and Michael Jones (Global Product Engineering)*

NEVRON

www.nevron.com

Nevron Data Visualization

The leading data visualization components for desktop and web development in a single package.



solutions available for

Microsoft ASP.net | Microsoft Silverlight | WPF | WinForms | SharePoint | Microsoft SQL Server

Learn more at **www.nevron.com** today

www.nevron.com | email@nevron.com | +1 888-201-6088 (Toll free, USA and Canada)

Visual Studio Partner

Creating a Location-Aware App with Geofencing

Tony Champion

The ever-increasing adoption of mobile devices is propelling the creation of location-aware apps. The opportunities for apps that know and react to a user's location are virtually limitless. Windows 8 included geolocation from the beginning, giving developers an easy way to determine the current location of a device. The Windows simulator even includes support for testing this feature. With Windows 8.1, these APIs have been extended with the concept of geofencing.

A geofence is a defined region around a GPS location that can be registered with Windows so an app can receive notifications when the device enters or leaves that region. Let's say you're a service technician with a dozen house calls to make in a given day. What if the app that manages your appointments could automatically send a text message to the home owner when you were five minutes away? Or suppose an amusement park wanted to announce the presence of characters, but only to people within a certain proximity of the character to limit the number of people that might show up. The possibilities are endless.

This article will explore the use of geofences in Windows 8.1. However, the geofencing API is also shared with Windows Phone 8.1, which means you can implement the same features on both platforms. You'll learn how to add geofences to your location-based app, plus how to handle events when the app is in the foreground and how to handle notifications when the app is in the background.

This article discusses:

- Adding geolocation support to a Windows Store app
- Determining user location
- Creating geofences
- Using geofences in the foreground
- Using geofences in the background

Technologies discussed:

Windows 8.1, Windows Store Apps

Code download available at:

msdn.microsoft.com/magazine/msdnmag0914

Adding Geolocation Support

Because geofences are part of the geolocation API in the Windows SDK, you must add support for this API to your app before you can use it. Luckily, Windows Store apps need very little setup to use geolocation. In fact, the only requirement is the package appxmanifest must include the Location capability. To add this, open the solution's appxmanifest in the designer and check "Location" in the capabilities tab, as shown in **Figure 1**.

Once this capability has been added to your project, the app is able to access Windows location services through the geolocation API if given permission by the user. This capability also adds a Permissions section to the app's Settings charm, which

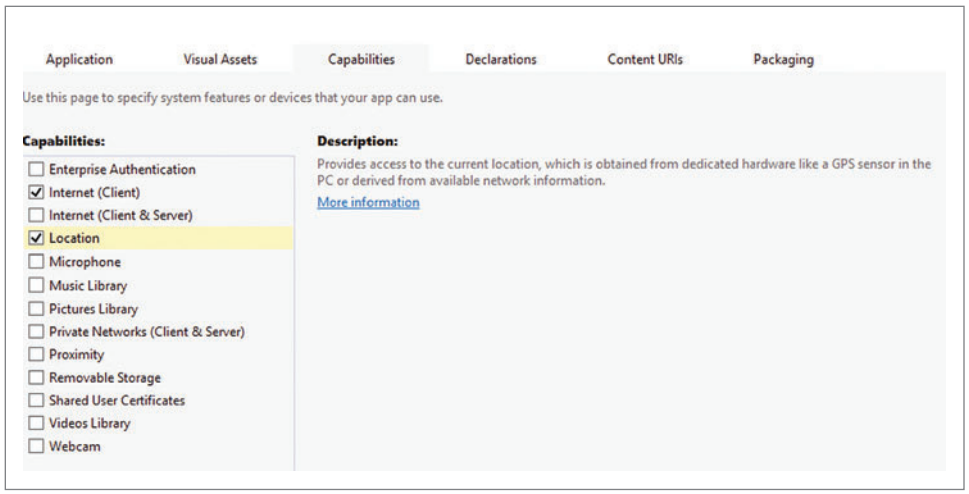


Figure 1 Enable **Location Capability**

lets the user enable and disable access to the device's location, as shown in **Figure 2**.

In addition to granting access to the location services on an individual app level, Windows can disable location services for the entire device. Location services are enabled by default in Windows, but users or systems administrators are free to change this by going to Control Panel | Hardware and Sound | Location Settings.

Because there are several different scenarios that can affect the app's access to the device's location, it's important your app know when there's a change to this access. The SDK lets the app monitor its access to the location services through an event exposed in the DeviceAccessInformation class in the Windows.Devices.Enumeration namespace. An instance of the DeviceAccessInformation class is created through a static method, CreateFromDeviceClass, which takes a DeviceClass enumeration that specifies the hardware device on which you want information. For the location services, this value is DeviceClass.Location. After creating an instance of DeviceAccessInformation, you can determine the current access level from the CurrentStatus property and listen for access changes through the AccessChanged event, like so:

```
DeviceAccessInformation deviceAccess =
  DeviceAccessInformation.
CreateFromDeviceClass(DeviceClass.Location);

DeviceAccessStatus currentStatus = deviceAccess.
CurrentStatus;
// Setup geolocation based on status

// Listen for access changes
deviceAccess.AccessChanged += deviceAccess_
AccessChanged;
```

The CurrentStatus property and AccessChanged event both receive a DeviceAccessStatus enumeration that describes the current access level. **Figure 3** shows the available values and their definitions.

Determining Location

If you're going to build a location-aware app that uses a geofence, you need to be able to determine the user's location. Though this is not necessarily required to implement geofences, you'll see later in this article that it comes in handy when validating geofence events.

The Windows SDK exposes a Geolocator class that attempts to determine the current GPS location of the device through several methods. The Geolocator and all location classes can be found in the Windows.Devices.Geolocation namespace. The first time you use a Geolocator class must be in the UI thread of the app, because Windows will prompt the user for permission to use the location services. Users are prompted only once per device and they can change this setting at any time by using the Permissions setting described earlier. Therefore, if a user blocks the original request or disables the permission from the Settings charm, you won't be able to programmatically change the access back and must rely on a UI notification to prompt users to change it themselves.

To determine the device's current location, Geolocator has a GetGeopositionAsync method that returns a Geoposition object. This object contains a Geocoordinate instance in the Coordinate property that contains the information returned by location services. Arguably the three most important pieces of information in a Geocoordinate class are the longitude, latitude and altitude of the current GPS position. In Windows 8.1, these are contained in the Point property, which is a Geopoint instance. The Latitude, Longitude and Altitude properties of the Geocoordinate class are there for backward compatibility, but the Point property should be used going forward.

There are several other valuable pieces of information in the Geocoordinate class that are helpful in dealing with geofences. The most important is the Accuracy property, which defines the GPS accuracy in meters. The Geolocator class uses the Windows location service to determine the current location. The location service uses several different approaches to do this. The most accurate method is when the device has a GPS radio that's enabled and receiving a signal. If that's not available, the location service tries using the device's Wi-Fi connection to pinpoint the location. Finally, it tries a less accurate approach using IP resolution if the device doesn't have an active Wi-Fi connection. Obviously, the accuracy of the current location is a significant factor in the validity of the geofences in your app and should be taken into consideration. For instance, IP resolution is only accurate to within 10 miles. That really won't help much if you're using geofences in an amusement park. With such

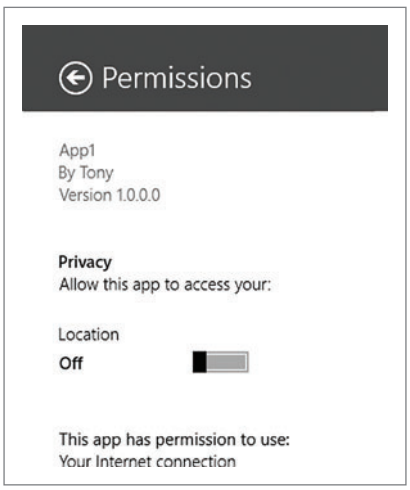


Figure 2 **Permissions Settings**

a wide range of accuracy for a device, it's important to consider the size of your geofence in comparison to the accuracy of the location to determine if the geofence is valid.

The Timestamp property is extremely useful in determining how old the data is. When dealing with geofences, timing can be very important. If you have geofences in your amusement park, for example, your app might use a different workflow if a user enters the parking lot during business hours versus the middle of the night. **Figure 4** shows an example of getting the current location of the user's device.

Creating Geofences

There are currently four different constructors for the Geofence class that allow you to define the location and behavior of a geofence. All of the geofencing structures are found in the Windows.Devices.Geolocation.Geofencing namespace. It's important to know how you want the Geofence instance to behave prior to creating it, because all of the exposed properties are read-only. Therefore, if you don't specify something during construction, you'll have to replace the geofence with a new one if you want to modify it.

The minimum information you need to create a Geofence class is a unique string identifier and a definition of the shape of the geofence, which is represented by an implementation of the IGeoshape interface. The only shape supported in Windows 8.1 is the Geocircle, which is defined by a center location and a radius in meters. As with the Geofence, these values must be defined at construction and can't be changed afterward. The radius can be anywhere from .1 meters to .25 of the earth's circumference, which should be sufficient to handle any size your app requires. The following example shows how to create a Geofence at the current device's location that has a 50-meter radius:

```
Geolocator geo = new Geolocator();

try {
  Geoposition pos = await geo.GetGeopositionAsync();

  Geocircle shape = new Geocircle(pos.Coordinate.Point.Position, 50.0);
  Geofence geofence = new Geofence("myUniqueId", shape);
}
catch (Exception) { }
```

Windows can monitor several different device interactions with a geofence. By default, it will monitor when a device enters and exits the defining area of the geofence. In addition, you can specify notification when a geofence has been removed from being monitored. It's a requirement that either the entering or exiting state be monitored, so you can't monitor only the removal of a geofence, which, to be honest, wouldn't be very useful to your app in the first place. Monitoring states can be defined using a combination of MonitoredGeofenceStates enumerations and are stored in the MonitoredStates property.

Figure 3 **AccessChanged Event**

```
private void deviceAccess_AccessChanged(
  DeviceAccessInformation sender, DeviceAccessChangedEventArgs args)
{
  switch (args.Status)
  {
    case DeviceAccessStatus.Allowed:
      // Access to the device is allowed
      break;
    case DeviceAccessStatus.DeniedByUser:
      // Denied by user when prompted or thru Permissions settings
      break;
    case DeviceAccessStatus.DeniedBySystem:
      // Denied at the system level from the Control Panel
      break;
    case DeviceAccessStatus.Unspecified:
      // Unknown reason for access to the device
      break;
  }
}
```

A geofence can also be a single- or multi-use entity. It's considered to be used once all of the monitored states have occurred. Therefore, if you specify the entered and exit states, the device must enter and then leave the specified area before the geofence is considered used. Unless otherwise specified, the monitoring will continue until the geofence is removed. The SingleUse property identifies whether the Geofence is set for single use. The following builds on the previous example and shows the second Geofence constructor:

```
MonitoredGeofenceStates monitor = MonitoredGeofenceStates.Entered |
                                  MonitoredGeofenceStates.Exited |
                                  MonitoredGeofenceStates.Removed;

bool singleUse = true;

Geofence geofence = new Geofence("myUniqueId", shape, monitor, singleUse);
```

By default, a device must remain on the monitored side of a boundary for 10 seconds before an app is notified. This keeps the system from firing multiple events if a device is right on the edge and moving back and forth. This value is the DwellTime and can be set to any TimeSpan greater than 0. The same DwellTime will be used for entering and exiting the area. Therefore, if you need different values, you'll have to create two geofences, one for entering and one for exiting. The following uses the third constructor by setting the DwellTime to 20 seconds:

```
TimeSpan dwellTime = new TimeSpan(0, 0, 20);
Geofence geofence = new Geofence("myUniqueId", shape, monitor,
  singleUse, dwellTime);
```

The final constructor allows you to set the StartTime and Duration of the geofence. A geofence will become active once the StartTime is in the past. By default, the StartTime is set to 0 or the beginning of a time handled in a DateTime class. If a geofence is created with a StartTime in the past and the device is already inside the defining area, the Entered state will be reported once the DwellTime has passed. In conjunction with the StartTime, you can define a Duration for how long the geofence will be active from the StartTime. If the Duration is set to 0 (the default value), the geofence remains active as long as it's registered to be monitored. When a Duration value has been set, the app will be notified of its expiration if the Removed monitor state is selected. Here's the final constructor, which sets a StartTime for midnight, Jan. 1, 2015, and a duration of 365 days:

```
DateTime startTime = new DateTime(2015, 1, 1);
TimeSpan duration = new TimeSpan(365, 0, 0, 0, 0);

Geofence geofence = new Geofence(
  "myUniqueId", shape, monitor, singleUse, dwellTime, startTime, duration);
```

Using Geofences in the Foreground

After a geofence has been created, the next step is to register it to be monitored so your app can receive notifications about it. This is handled through a GeofenceMonitor instance. Each app will have a single GeofenceMonitor that can be accessed through the static property of GeofenceMonitor.Current. The GeofenceMonitor

maintains a list of all registered geofences in the Geofences property, which is an IList<Geofence>. Adding and removing geofences from this list is as easy as using the IList methods you're used to, such as Add and Remove. Once an app registers a geofence, it's persisted to disk. This means you only need to register the geofence once, even between app uses. If you attempt to register a geofence with a duplicate id, an error will be generated, so it's a good policy to verify the id isn't present prior to registration:

```
IEnumerable<Geofence> existing =
  GeofenceMonitor.Current.Geofences.Where(g => g.Id == geofence.Id);

if (existing.Count() == 0)
{
  GeofenceMonitor.Current.Geofences.Add(geofence);
}
else
{
  // Handle duplicate entry
}
```

Once the geofence has been added to the GeofenceMonitor, you'll receive notifications about it based on the monitor states that have been selected. These notifications are handled through the GeofenceStateChanged event of the GeofenceMonitor:

```
GeofenceMonitor.Current.GeofenceStateChanged += GeofenceStateChanged;
```

When the GeofenceStateChanged event is fired, the affected geofences are not sent in the args property, which is common for most changed event handlers. To get the notification about what has changed, you call the ReadReports method of the current GeofenceMonitor. This will return a collection of recent monitor notifications, sorted in a descending timestamp order, that have happened for the app. Each notification is represented by a GeofenceStateChangeReport class.

The GeofenceStateChangeReport has a Geofence property that references the geofence whose state has changed, and a Geoposition property that provides the location of the device responsible for the state being changed. It also has a NewState property, which is a GeofenceState enum that identifies which monitoring state was triggered. Finally, there's a RemovalReason property, which is a GeofenceRemovalReason enum that can be either Used or Expired. The default is Used for any event and doesn't mean the geofence has been removed. It simply provides a removal reason if the NewState value is Removed.

The number of reports returned by ReadReports is controlled by Windows. There's no guarantee how many reports will be stored or for how long, so if you need to maintain any information, you'll need to keep a separate copy in the app. **Figure 5** is an example of handling the GeofenceStateChanged event.

Using Geofences in the Background

One of the additional issues developers have to take into consideration is the lifecycle of a Windows Store app. If an app isn't visible on the screen, Windows suspends the application while keeping it in memory, in order to help preserve performance and battery life. This means your app will no longer be running. In most cases this isn't a cause for concern because if the user isn't directly interacting with the app, there's nothing for the app to do.

However, in some scenarios, an app still needs the ability to continue to perform certain tasks, even if the app isn't running. Windows addresses this with the concept of background tasks. A background task is a small unit of code that executes in response to a predefined system event. There are more than a dozen different events your app can register a background task for, and listening to geofences is one of them.

Adding a Background Task

All background tasks are implemented by creating sealed classes that implement the IBackgroundTask interface located in the Windows.ApplicationModel.Background namespace. This interface defines a single Run method that must be implemented. Any classes that implement IBackgroundTask must exist in a Windows Runtime Component and won't execute if they're part of your main app project. It's common practice to put all background tasks for an app in a single Windows Runtime Component.

> In some scenarios an app still needs the ability to continue to perform certain tasks, even if the app isn't running.

To create a background task, add a Windows Runtime Component project to your solution. Add a reference to that new project to your main app project so when you register the task the app has visibility to the class structure. **Figure 6** illustrates a background task that responds to geofence state changes.

Registering a Background Task

Once you've created a background task, the next step is to register that background task with Windows. As with location services, the user must grant your app permission to use background tasks. You can prompt the user via the static method RequestAccessAsync of the BackgroundExecutionManager class. If the user has already answered the prompt, this method will return the current status of running background tasks on the app.

The actual registration is completed through an instance of BackgroundTaskBuilder. It requires three pieces of information to be valid. The first is a unique name for the app. The second

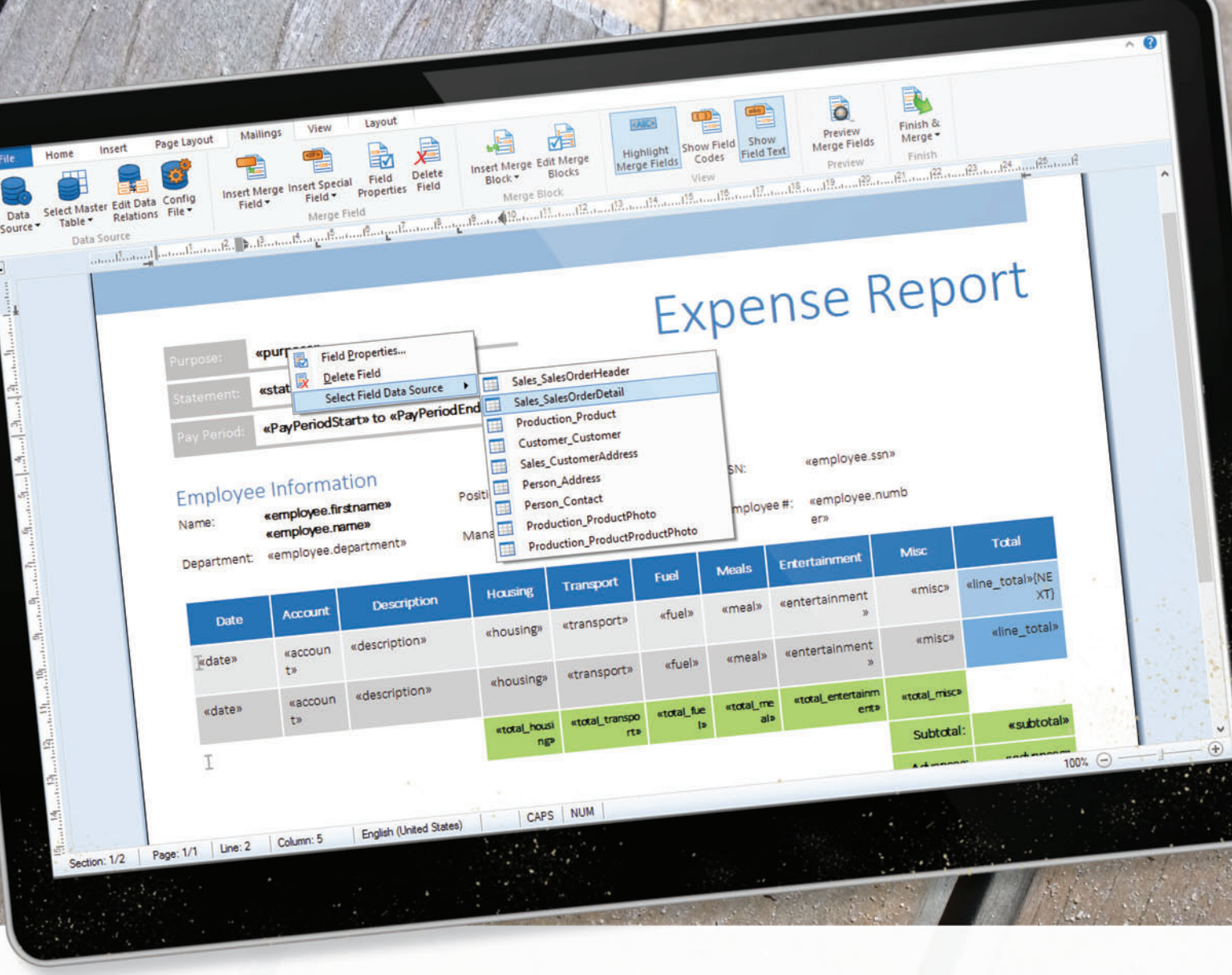
Figure 4 **Getting Geoposition**

```
Geolocator geo = new Geolocator();

try
{
  Geoposition pos = await geo.GetGeopositionAsync();

  double longitude = pos.Coordinate.Point.Position.Longitude;
  double latitude = pos.Coordinate.Point.Position.Latitude;
  double altitude = pos.Coordinate.Point.Position.Altitude;
  double accuracy = pos.Coordinate.Accuracy;
  DateTimeOffset timestamp = pos.Coordinate.Timestamp;

}
catch (Exception ex)
{
  // Handle errors like unauthorized access to location services
}
```



Check out the new trend in reporting:

FLOW TYPE LAYOUT REPORTING BY TEXT CONTROL

Text Control Reporting combines the power of a reporting tool and an easy-to-use WYSIWYG word processor - fully programmable and embeddable in your .NET **Desktop**, **Web** and **Mobile** applications.

- MS Word compatible
- Create Adobe PDF files
- 2D and 3D charting
- 1D and 2D barcodes

Download your 30-day trial version today:
www.textcontrol.com/reporting

Visual Studio Partner

the new TEXT CONTROL

Figure 5 **GeofenceStateChanged Event Handler**

```
void Current_GeofenceStateChanged(GeofenceMonitor sender, object args)
{

  IReadOnlyList<GeofenceStateChangeReport> reports =
    GeofenceMonitor.Current.ReadReports();

  foreach (GeofenceStateChangeReport report in reports)
  {
    switch (report.NewState)
    {
      case GeofenceState.Entered:
        // Handle entered state
        break;
      case GeofenceState.Exited:
        // Handle exited state
        break;
      case GeofenceState.Removed:
        if (report.RemovalReason == GeofenceRemovalReason.Used)
        {
          // Removed because it was single use
        }
        else
        {
          // Removed because it was expired
        }
        break;
    }
  }
}
```

is the TaskEntryPoint property that takes the full name of the background task class in a string, including the namespace.

The last piece of information defines the type of event for which you want to register. This is done by creating a trigger and using the SetTrigger method of the BackgroundTaskBuilder. Every background task can have just a single trigger. If you want to use the same background task for multiple triggers, you must create and register multiple background tasks. In order to track geofence changes, create a LocationTrigger instance and pass in a LocationTriggerType.Geofence enum value into the constructor. Currently geofencing is the only location-based trigger offered in Windows. The end result is that the Run method of the IBackgroundTask you created will be called every time there's a change to the state of a registered geofence. **Figure 7** demonstrates how to register a geofence background task.

The final step to enable your app to run background tasks is to add a Background Task declaration in the package appxmanifest. Inside the appxmanifest designer, select the Declarations tab, select Background Task from the Add Declarations dropdown, then click the Add button. In the details pane, select Location as the property and for the Entry point, put the classname of the TaskEntryPoint of your BackgroundTaskBuilder. Once you do this, you'll see a red "X" on the Application tab. Some background tasks require your app to be added to the lock screen in order for the tasks to run, and a location background task is one such task. The first step is to set the Lock screen notifications on the Application pane to either Badge or Badge and Tile Text. The next step is for the user to add the app to his lock screen. The BackgroundExecutionMananger.RequestAccessAsync method discussed earlier will add the app to the lock screen if the user approves it. However, you need to write code to notify the user in the event the user removes the app from the lock screen at a later date, because Windows only prompts the user to add the app a single time.

Figure 6 **Background Task**

```
public sealed class MyBackgroundTask : IBackgroundTask
{
  public void Run(IBackgroundTaskInstance taskInstance)
  {
    BackgroundTaskDeferral deferral = taskInstance.GetDeferral();

    var reports = GeofenceMonitor.Current.ReadReports();

    foreach (var report in reports)
    {
      // Handle each report
    }

    deferral.Complete();
  }
}
```

Figure 7 **Register Background Task**

```
private async void RegisterBackgroundTask(object sender, RoutedEventArgs e)
{
  BackgroundAccessStatus accessStatus =
    await BackgroundExecutionManager.RequestAccessAsync();

  if(accessStatus ==
    BackgroundAccessStatus.AllowedMayUseActiveRealTimeConnectivity ||
   accessStatus ==
     BackgroundAccessStatus.AllowedWithAlwaysOnRealTimeConnectivity)
  {

    BackgroundTaskBuilder taskBuilder = new BackgroundTaskBuilder();
    taskBuilder.Name = "MyGeoBackground";
    taskBuilder.TaskEntryPoint = "WindowsRuntimeComponent1.MyBackgroundTask";

    LocationTrigger trigger = new LocationTrigger(LocationTriggerType.Geofence);

    taskBuilder.SetTrigger(trigger);

    taskBuilder.Register();
  }

}
```

Your app will now respond to geofence state changes in the background. It's important to remember that this will take place even if your app is currently running, so the workflow of your app should take that into consideration if you're also monitoring geofences in the foreground for UI updates.

For a deeper description of background tasks in Windows 8.1, please refer to "Supporting Your App with Background Tasks (XAML)" at bit.ly/1i9AH8X.

Wrapping Up

Geofences add a new dynamic to any location-aware app, by letting your app respond to changes in proximity to predefined GPS coordinates. This allows you to easily provide hot spots your app can interact with around the globe. Not only can your app respond to these geofences when it's running, but by utilizing background tasks it can still respond to them when the app has been suspended or isn't even running. With smaller GPS-enabled devices becoming ever more popular, adding location-aware features to your app can provide a great UX, and I, for one, can't wait to see what gets created next. ■

Tony Champion *is president of Champion DS, is a Microsoft MVP, and is active in the community as a speaker, blogger, and author. He maintains a blog at tonychampion.net and can be reached at tony@tonychampion.net.*

Thanks to the following Microsoft technical expert for reviewing this article: *Robert Green*

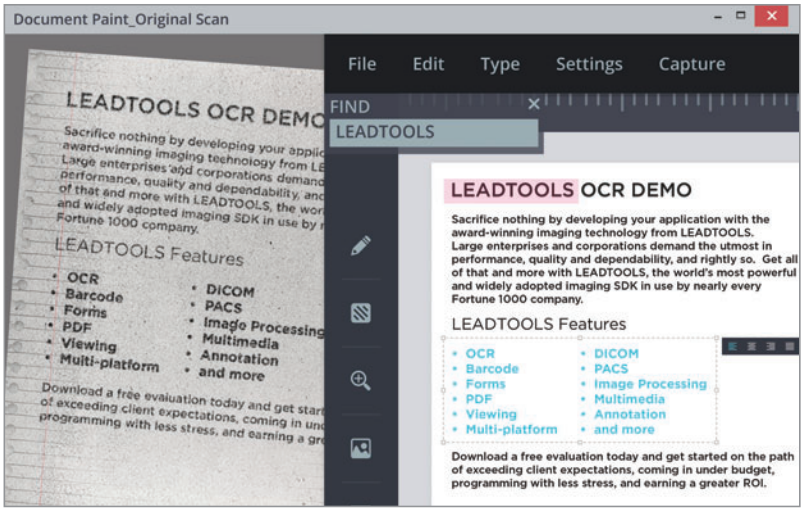
LEADTOOLS®
THE WORLD LEADER IN IMAGING SDKs

Best of the Best: LEADTOOLS Imaging SDKs

Choosing a software development kit for your application can be a difficult task. You must navigate a harmony of features, ease of use, support, cost and more in an equally hard-to-balance time frame. LEADTOOLS is the World Leader in Imaging SDKs and there are many reasons why.

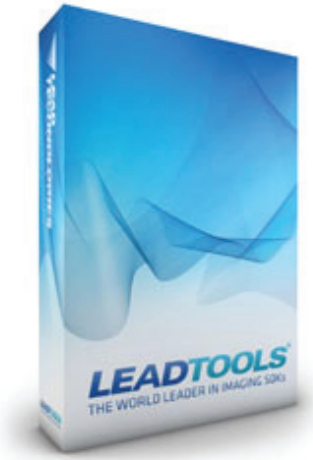
Unparalleled Experience

Since 1990, LEAD Technologies has been supplying imaging technology to software developers, integrators, contractors and solution providers throughout the United States government, military and most Fortune 1000 companies. LEAD's flagship SDKs, LEADTOOLS, are the most comprehensive and widely used imaging toolkits and are regarded as the gold standard across the globe. Any programmer writing an application for document, medical, multimedia, raster or vector imaging can benefit from the vast development experience packed into the millions of lines of code that comprise LEADTOOLS.



One Stop Shop

If you could summarize LEADTOOLS in one word it would probably be comprehensive. Why? It boasts the most imaging categories on the widest gamut of development platforms including .NET, Win32/64, WinRT, Windows Phone, HTML5/JavaScript, iOS, OS X, Android, Linux and more. With LEADTOOLS, you will be able to create astonishing applications featuring OCR, Forms, Barcode, PDF, DICOM, PACS, Image Processing, MPEG-2 Transport Stream, Multimedia Playback/Capture/Conversion and much more on virtually any platform you can imagine.



Tomorrow's Technology Today

LEAD Technologies is always at the forefront of imaging technology. This means that you can rest assured that when new standards and development platforms are released that LEADTOOLS will have something for you. LEAD also forms strategic partnerships with customers looking for tools targeting cutting edge technology that provide a dual benefit of a custom-tailored solution and expedited time-to-market.

Personal Touch

We software developers sometimes get a bad rap as being anti-social cave dwellers, but we're people too! LEAD Technologies gets that and has utilized the customer feedback and suggestions over the decades to make LEADTOOLS incredibly programmer-friendly. On top of that, free technical support over email, chat and phone is available and LEAD's developer support team loves to help you get the most out of LEADTOOLS with custom-made example projects and pointers to help you along the way.

To learn more please visit our website → **www.leadtools.com**

Developing Your First Game with Unity and C#, Part 2

Adam Tuliper

Welcome back to my series on Unity. In the first article, I covered some Unity basics and architecture. In this article, I'm going to explore 2D in Unity, which builds upon the 2D support Unity added in version 4.3. You could do 2D in Unity before 4.3, but the process was quite painful without a third-party toolkit. What I want is to just drag and drop an image into my scene and have it appear and work as I'd expect via a drag/drop interface. That's some of what Unity 4.3 brings to the table and in this article, I'll discuss more of its features while developing a basic 2D platformer game to learn some essential Unity concepts.

2D in Unity

To get 2D support in Unity, when creating a new project you select 2D from the dropdown in the new project dialog. When you do, project defaults are set to 2D (viewed under Edit | Project Settings | Editor) and any images imported into your project are brought in as sprites as opposed to just textures. (I'll cover this in the next section.) Also, the scene view defaults to 2D mode. This really just provides a helper button that fixes you to two axes during scene development, but has no effect in your actual game. You can click it at any time to pop in and out of 2D working mode. A 2D game in Unity is really still a 3D environment; your work is just constrained to the X and Y axes. **Figure 1** and **Figure 2** show the 2D mode selected and not selected. I have the camera highlighted so you can see an outline of the camera's viewing area, but note that it looks out into space as a rectangle shape.

This article discusses:

- Working with 2D in Unity
- Sprites
- Common functionality, including colliders, score-tracking and animation
- Creating prefabs for reuse

Technologies discussed:

Unity, C#, Microsoft .NET Framework

Code download available at:

bit.ly/UnityPlatformer

A 2D game in Unity is really still a 3D environment; your work is just constrained to the X and Y axes.

The highlighted camera is set up as an orthographic camera, one of two camera modes in Unity. This camera type, which is commonly used in 2D, doesn't scale objects further away as your eyes would see them; that is, there's no depth from the camera position. The other camera type is perspective, which shows objects as our eyes see them, with depth. There are various reasons to use one camera type instead of the other, but in general, choose

perspective if you need visual depth, unless you want to scale your objects accordingly. You can change the mode simply by selecting the camera and changing the projection type. I recommend trying this out and seeing how your camera's viewing area changes when you start moving objects further away into the Z axis. You can change the default behavior mode at any time, which affects only future importing of images into your project.

> When 3D is selected as the default behavior mode, images are recognized as type Texture.

If you have an existing project in Unity or aren't sure if you've selected 2D from the project dialog, you can set your project defaults for 2D by going to Edit | Project Settings | Editor; otherwise, you'll have to manually set the texture type on every 2D image you import, which is a bit tedious if you have a lot of artwork.

It's All About the Sprite

When 3D is selected as the default behavior mode, images are recognized as type Texture. You can't drag a texture into your scene; a texture must be applied to an object. This way isn't much fun to create 2D games. I want to just drag and drop an image and have it appear in my scene. If your default behavior mode is 2D, though, things get easy. When I drag and drop an image into Unity, it's now recognized as type Sprite.

This allows you to simply drag and drop your artwork into Unity, and then from Unity drag and drop it into your scene to build out your game. If you have artwork that looks small, rather than rescale it everywhere you can just decrease the Pixels To Units value. This is a very common operation in Unity for both 2D and 3D and typically is more performant than scaling objects via its transform's scale property.

When dropping objects, you might notice that one object ends up on top of another. Unity creates a series of vertices behind the scenes, even for 2D images, so the drawing order can differ on various parts of the images. It's always best to explicitly specify the z-order of your images. You can do this via three methods, listed in the order Unity draws your sprites:

1. Setting the "Sorting Layer" property in the Sprite Renderer.
2. Setting the "Order in layer" property, also on the Sprite Renderer.
3. Setting the Transform's Z position value.

The sorting layer takes priority over all, followed by order in layer, which in turn takes priority over the transform's z value.

Sorting layers draw in order of definition. When you add other layers (in Edit | Project Settings | Tags and Layers), Unity draws out any object it finds on the Default layer first (then Order in Layer takes over, and then the Transform's Z position value) , then Background, then Platforms, and so forth. So you can easily fix overlapping images by setting them to the Platforms layer and giving the one you want on top an Order in Layer of 1, so it will be drawn after anything with Order in Layer of 0.

Common Functionality

Figure 3 shows a level containing some platforms and background images that were built out by dragging and dropping and setting the sorting layers.

As it stands now, it looks like a game, but it doesn't act like one. At a minimum, it needs a number of features to be a functional game. I'll discuss several of these in the following sections.

Keyboard, Mouse and Touch Movement In Unity, keyboard, mouse, accelerometer and touch are read through the input

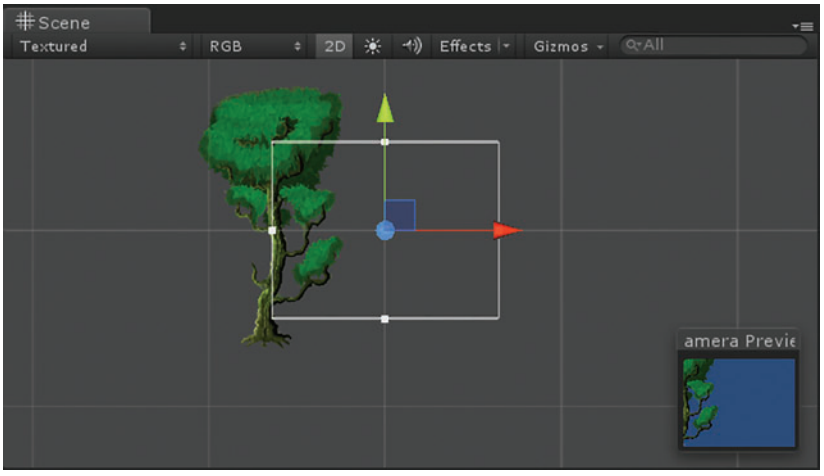


Figure 1 **2D Mode Selected—Camera Has Focus**

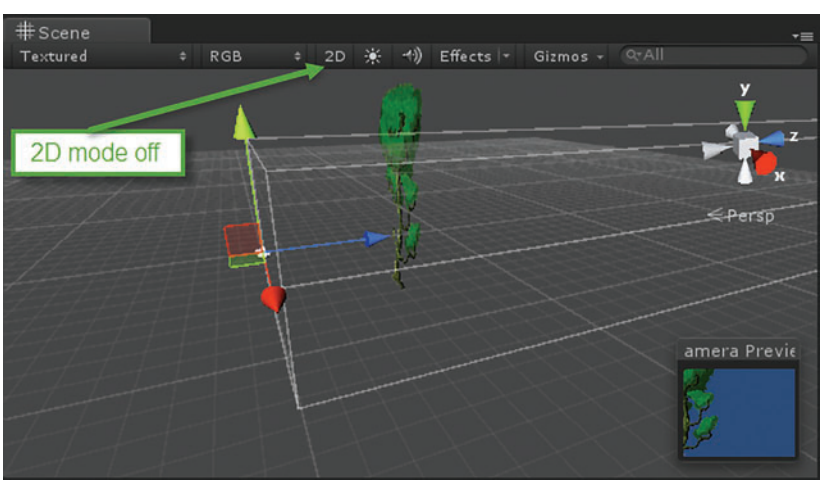


Figure 2 **2D Mode Not Selected—Camera Has Focus**

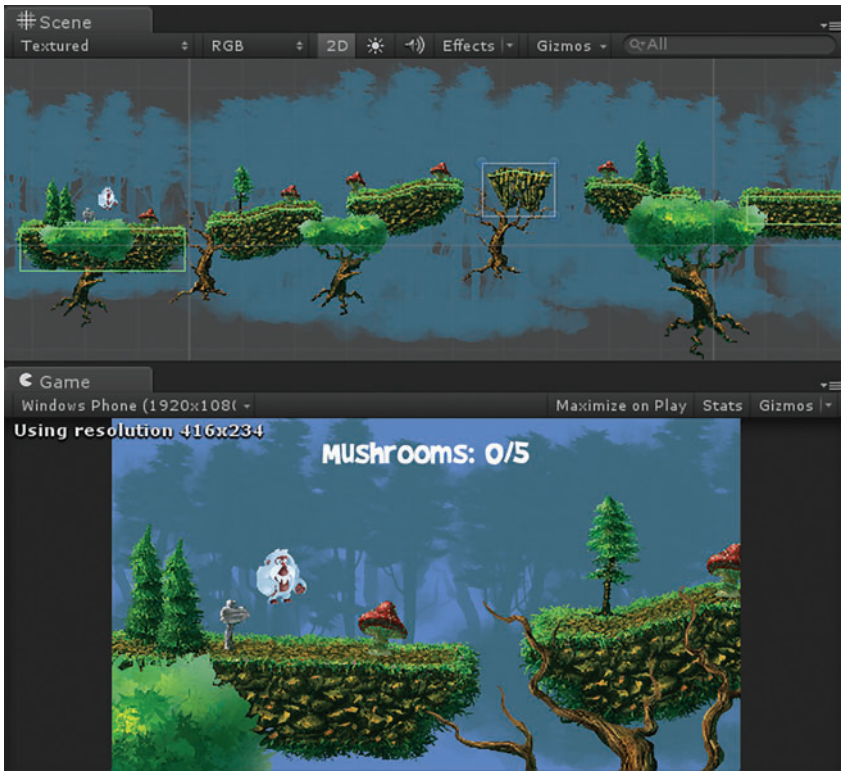


Figure 3 **A Game Level**

system. You can read input movement and mouse clicks or touch easily using a script like the following on the main player (I'll build on this script shortly.):

```
void Update()
{

  // Returns -1 to 1
  var horizontal = Input.GetAxis("Horizontal");

  // Returns true or false. A left-click
  // or a touch event on mobile triggers this.
  var firing = Input.GetButtonDown("Fire1");

}
```

If you check Edit | Project Settings | Input, you can see the default inputs (Unity gives you a bunch that are there in every new project) or set new ones. **Figure 4** shows the defaults for reading horizontal movement. The "left" and "right" settings represent the left and right arrow keys but notice also that "a" and "d" are used for horizontal movement. These can map to joystick inputs. You can add new ones or change the defaults. The Sensitivity field controls how fast Unity will go from 0 to 1 or -1. When the right arrow is pressed, the first frame might yield a value of .01 and then scale pretty quickly up to 1, although you can adjust the speed to give your character instant horizontal speed or movement. I'll show you the code shortly for applying these values to your game

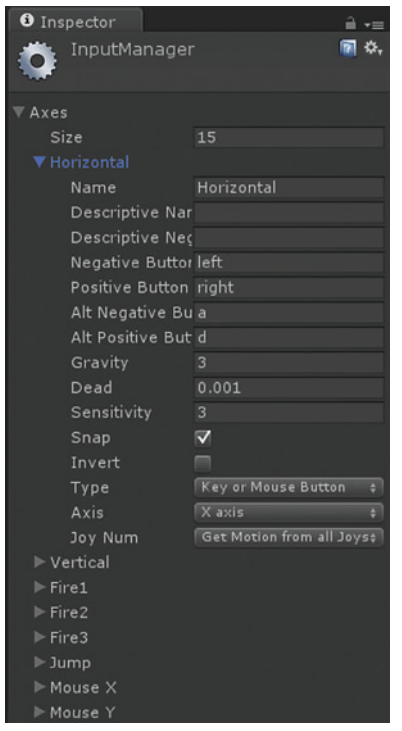


Figure 4 **Horizontal Input Defaults**

objects. There's no actual GameObject component required for reading these values; you simply use the Input keyword in your code to access the functionality for reading input. Input, as a general rule, should be read in the Update function as opposed to FixedUpdate to avoid missing input events.

Linear Movement Things need to be able to move. If this is a top-down game, gravity typically isn't important. If it's a platformer, gravity is exceedingly important. In either case, object collision detection is critical. Here are the basic rules. A Rigidbody2D or RigidBody (used for 3D) component added to a game object will automatically give that component mass and make it understand gravity and receive forces. According to Wikipedia, "In physics, a rigid body is an idealization of a solid body in which deformation is neglected. In other words, the distance between any two given points of a rigid body remains constant in time regardless of external forces exerted on it." The same principle applies in games. Adding a rigid body lets you make calls like the ones in **Figure 5**.

As a general rule, linear movement should happen via Update and accelerated movement via FixedUpdate. If you're a beginner, it might seem confusing what to use when, and, in fact, linear movement will work in either function. But you'll get better visual results by following this rule.

Collision Detection An object gets its mass from its RigidBody component, but you also need to tell Unity how to handle collisions with this object. The size and shape of your artwork or models doesn't matter here, although scaling does affect physics on the object itself. What matters is the size and shape of the collider component, which is simply a defined region around, on or within the object that you want Unity to detect another object contacting. This is what enables scenarios like detecting when you enter the region of an idle zombie or boulders that go bouncing down a mountain side on approach.

There are variously shaped colliders. A collider for 2D can be a circle, edge, polygon or box. Box colliders are great for objects shaped like squares or rectangles, or when you simply want to detect collisions in a square area. Think of a platform you can stand on—this is a good example of a box collider. Simply adding this component to your game object allows you to

Figure 5 **Adding Movement and Velocity**

```
void FixedUpdate()
{
  // -1 to 1 value for horizontal movement
  float moveHorizontal = Input.GetAxis("Horizontal");

  // A Vector gives you simply a value x,y,z, ex  1,0,0 for
  // max right input, 0,1,0 for max up.
  // Keep the current Y value, which increases for each interval because of gravity.
  var movement = new Vector3(moveHorizontal * _moveSpeed, rigidbody.velocity.y, 0);
  rigidbody.velocity = movement;

  if (Input.GetButtonDown("Fire1"))
  {
    rigidbody.AddForce(0, _jumpForce, 0);
  }
}
```

take advantage of physical collisions. In **Figure 6**, I added a circle collider and rigid body to the character and a box collider to the platform. When I click play in the Editor, the player immediately drops onto the platform and stops. No code required.

You can move and resize the region a collider covers by changing its properties on the collider component. By default, objects with colliders don't pass through each other (except for triggers, which I'll cover next). Collisions require collider components on both game objects and at least one object has to have a RigidBody component, unless it's a trigger.

An object gets its mass from its RigidBody component, but you also need to tell Unity how to handle collisions with this object.

If I want Unity to call my code when this collision instance first happens, I simply add the following code to a game object via a script component (discussed in the previous article):

```
void OnCollisionEnter2D(Collision2D collision)
{
  // If you want to check who you collided with,
  // you should typically use tags, not names.
  if (collision.gameObject.tag == "Platform")
  {
    // Play footsteps or a landing sound.
  }
}
```

Figure 6 **Adding Colliders**

Triggers Sometimes you want to detect a collision but don't want any physics involved. Think of a scenario like picking up treasure in a game. You don't want the coins to be kicked out in front of the player when it approaches; you want the coins to be picked up and not interfere with player movement. In this case, you use a collider called a trigger, which is nothing more than a collider with the IsTrigger checkbox enabled. That turns off the physics and Unity will *only* call your code when object A (which contains a collider) comes within the region of object B (which also has a collider). In this case, the code method is OnTriggerEnter2D instead of OnCollisionEnter2D:

```
void OnTriggerEnter2D(Collider2D collider)
{
  // If the player hits the trigger.
  if (collider.gameObject.tag == "Player")
  {
    // More on game controller shortly.
    GameController.Score++;
    // You don't do: Destroy(this); because 'this'
    // is a script component on a game object so you use
    // this.gameObject or just gameObject to destroy the object.
    Destroy(gameObject);
  }
}
```

The thing to remember is that with triggers, there's no physical interaction, it's basically just a notification. Triggers do not require a Rigidbody component on the game object, either, because no force calculations are taking place.

One thing that often trips up new developers is the behavior of rigid bodies when you add colliders to them. If I have a circle collider on my object and I put this object on an inclined plane (as indicated by its collider shape), it starts to roll, as **Figure 7** shows. This models what you'd see in the physical world if a wheel was set on an incline. The reason I don't use a box collider for my character is because a box has edges that can get caught up when moving over other colliders' edges, yielding a less smooth experience. A circle collider makes this smoother. However, for times when a smooth rotation isn't acceptable, you can use the Fixed Angle setting on the Rigidbody component.

Audio To hear sound, you need an Audio Listener component, which already exists on any camera by default. To play a sound, simply add an Audio Source component to a game object and set the audio clip. Unity supports most major audio formats and will

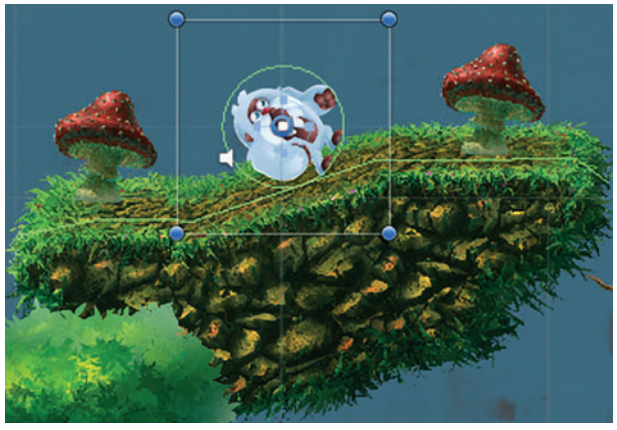


Figure 7 **Using a Circle Collider for Smooth Movement**

Figure 8 **Creating the _scoreText Variable**

```
public class GameController : MonoBehaviour
{
  private int _score;

  // Drag a GuiText game object in the editor onto
  // this exposed field in the Editor or search for it upon startup
  // as done in Figure 12.
  [SerializeField]
  private GUIText _scoreText;

  void Start()
  {
    if (_scoreText == null)
    {
      Debug.LogError("Missing the GuiText reference. ");
    }
  }

  public int Score
  {
    get { return _score; }
    set
    {
      _score = value;
        // Update the score on the screen
      _scoreText.text = string.Format("Score: {0}", _score);
    }
  }
}
```

encode longer clips to MP3. If you have a bunch of audio sources with clips assigned in the Unity Editor, keep in mind they'll all be loaded at run time. You can instead load the audio via code located in a special resource folder and destroy it when done.

When I imported audio into my project, I kept it as a WAV file, which is uncompressed audio. Unity will re-encode your longer audio to optimize it, so always use the best quality audio you have. This is especially true for short files like sound effects, which Unity won't encode. I also added an Audio Source component to my main camera, though I could've added it to any game object. I then assigned the Adventure audio clip to this Audio Source component and checked off Loop, so it constantly loops. And in three simple steps, I now have background music when my game plays.

GUI/Heads-Up Display A GUI system can comprise many things in a game. It may involve the menu system, the health and score display, weapons inventory, and more. Typically, a GUI system is what you see on the screen that stays put no matter where the camera is looking (although it doesn't have to). The Unity GUI functionality is currently undergoing a complete revision and the new uGUI system is coming out in Unity 4.6. Because that isn't released yet, I'll simply discuss some of the basic functionality here, but check out my channel9 blog for details of the new GUI system at channel9.msdn.com/Blogs/AdamTuliper.

To add simple display text to the screen (for example, score: 0), I clicked on Game Object | Create Other | GUI Text. This option no longer exists in Unity 4.6, so you'll want to watch that video on uGUI I mentioned. You can still add a GUI Text component to the game object in 4.6 by clicking the Add Component button; it's just missing from the Editor menu. With the existing (legacy) Unity GUI system, you can't see your GUI objects in the scene view, only in the Game view, which makes layout creation a little weird. If you like, you can use pure code to set up your GUI, and there's a GUILayout class that lets you track widgets automatically. But I prefer a GUI system where I can click and drag and work with easily, which is why I find uGUI far superior. (Before uGUI, the leader in this area was a pretty solid third-party product called NGUI, which was actually used as the initial code base for uGUI.)

The easiest way to update this display text is to simply search for or assign in the Editor a reference to the GUI Text game object, and treat it like a label in .NET and update its text property. This makes it easy to update the screen GUI text:

```
void UpdateScore()
{
  var score = GameObject.Find("Score").GetComponent<GUIText>();
  score.text = "Score: 0";
}
```

This is a slightly shortened example. For performance, I'd actually cache a reference to the GUIText component in the Start method so as to not query for it every method call.

Score Tracking Tracking scores is easy. You simply have a class that exposes a public method or properties on which to set a score. It's common in games to have an object called a Game Controller that acts as the game's organizer. The Game Controller can be responsible for triggering game saves, loading, score keeping and more. In this example, I can simply have a class that exposes a score variable, as shown in **Figure 8**. I assign this component to an empty game object, so it's available when the scene loads. When the score is updated, the GUI is in turn updated. The _scoreText variable is assigned in the Unity editor. Just drop any GUIText game object onto this exposed field or use the search widget where this script component exposes the Score Text variable in the editor.

I can then simply update (in this example) the mushroom's trigger code as follows to increment the score with each pickup:

```
void OnTriggerEnter2D(Collider2D collider)
{
  if (collider.gameObject.tag == "Player")
  {
    GameController.Score++;
    Destroy(gameObject);
  }
}
```

Animations Just as with XAML, animations are created by carrying out various actions in key frames. I could easily devote a whole article just to animations in Unity, but I'll keep it brief here because of space. Unity has two animation systems, the legacy system and the newer Mecanim. The legacy system uses animation(.ani) files, while Mecanim uses states to control which animation file plays.

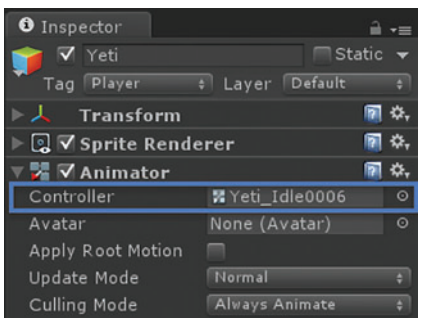


Figure 9 **The Animator Component Pointing to a Controller**

Animation in 2D uses Mecanim by default. The simplest way to create an animation is to drag and drop images into your scene and let Unity create the animations for you. To start, I drag some single sprites into Unity and in turn Unity creates several things for me. First, it creates

Spreadsheets Made Easy.

Fastest Calculations

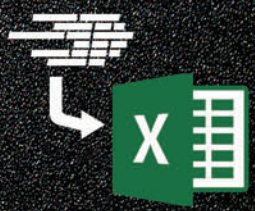
Evaluate complex Excel-based models and business rules with the fastest and most complete Excel-compatible calculation engine available.

Powerful Controls

Add powerful Excel-compatible viewing, editing, formatting, calculating, filtering, sorting, charting, printing and more to your WinForms, WPF and Silverlight applications.

Comprehensive Charting

Enable users to visualize data with comprehensive Excel-compatible charting which makes creating, modifying, rendering and interacting with complex charts easier than ever before.



Scalable Reporting

Easily create richly formatted Excel reports without Excel from any ASP.NET, Windows Forms, WPF or Silverlight application.

Download your free fully functional evaluation at SpreadsheetGear.com

SpreadsheetGear

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

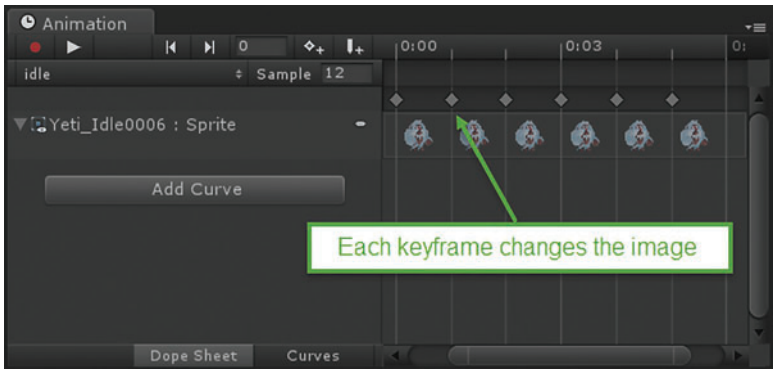


Figure 10 **The Idle Animation Data**

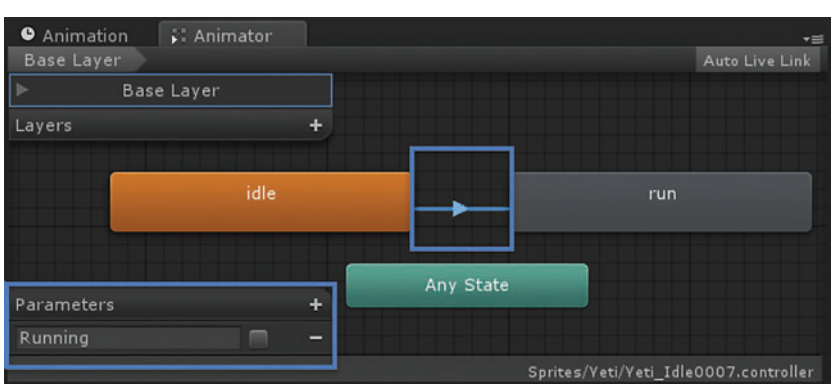


Figure 11 **Changing from the Idle to Run States**

a game object with a sprite renderer component to draw the sprites. Then it creates an animation file. You can see this by going to Window | Animator and highlighting your game object. The animator shows the animation file assigned, which, in my case, contains six key frames because I dropped six images into my scene. Each key frame controls one or more parameters on some component; here, it changes the Sprite property of the Sprite Renderer component. Animations are nothing more than single images showing at some rate that makes the eye perceive movement.

Next, Unity creates an Animator component on the game object, as shown in **Figure 9**.

This component points to a simple state machine called an animation controller. This is a file Unity creates, which just shows the default state; in other words, it's always in the "idle" state as that's the only state available. This idle state does nothing more than point to my animation file. **Figure 10** shows the actual key frame data on the time line.

This might seem like a lot to go through just to play an animation. The power of state machines, though, is that you can control them by setting simple variables. Remember, a state does nothing more than point to an animation file (although in 3D you can get fancy and do things like blend animations together).

I then took more images to make a run animation and dropped them onto my Yeti game object. Because I already have an animator component on the game object, Unity just creates a new animation file and adds a new state called "run." I can simply right-click on idle and create a transition to run. This creates an arrow between the idle and run states. I can then add a new variable called "Running," which is simple to use—you just click on the arrow between the states and change the condition to use the variable, as shown in **Figure 11**.

When Running becomes true, the idle animation state changes to the run animation state, which simply means the run animation file plays. You can control these variables in code very easily. If you want to start your run animation by triggering the run state when the mouse button is clicked, you can add the code shown in **Figure 12**.

In my example, I used single sprites to create an animation. It's pretty common, though, to use a sprite sheet—a single image file with more than one image in it. Unity supports sprite sheets, so it's a matter of telling Unity how to slice up your sprite, and then dropping those slices into your scene. The only steps that are different are changing Sprite Mode from Single to Multiple on the sprite properties, and opening the Sprite Editor, which can then automatically slice the sprite and apply the changes, as shown in **Figure 13**. Finally, you expand the sprite (there's a little arrow on the sprite's icon in the project view), highlight the resulting sprites, and drop them into your scene as you did earlier.

Animation can be a complicated subject until you learn the system. For more information, check out my channel9 blog or one of the many fine resources on Unity's learning site.

The End of the Level When the player gets to the end of the level, you can simply have a collider that's set to a trigger and

Figure 12 **Changing State with Code**

```
private Animator _animator;
void Awake()
{
    // Cache a reference to the Animator component from this game object
    _animator = GetComponent<Animator>();
}
void Update()
{

  if (Input.GetButtonDown("Fire1"))
  {
    // This will cause the animation controller to
    // transition from idle to run states
    _animator.SetBool("Running", true);
  }
}
```

Additional Learning

See this project built out: channel9.msdn.com/Events/Build/2014/2-503

Microsoft Virtual Academy - Developing 2D & 3D Games with Unity for Windows: aka.ms/UnityMVA

Adam's Channel 9 Blog: aka.ms/AdamChannel9

Unity Resources: unity3d.com/learn

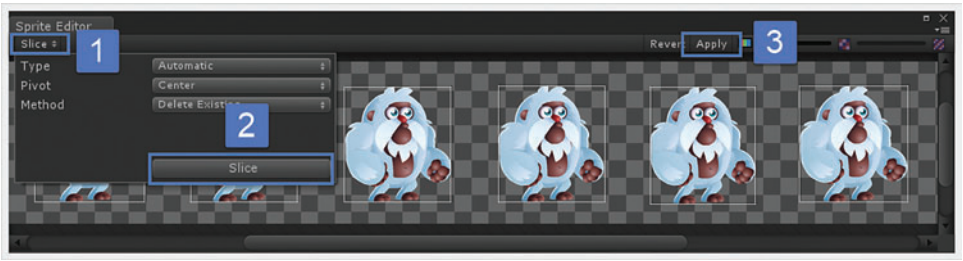


Figure 13 **Creating a Sprite Sheet**

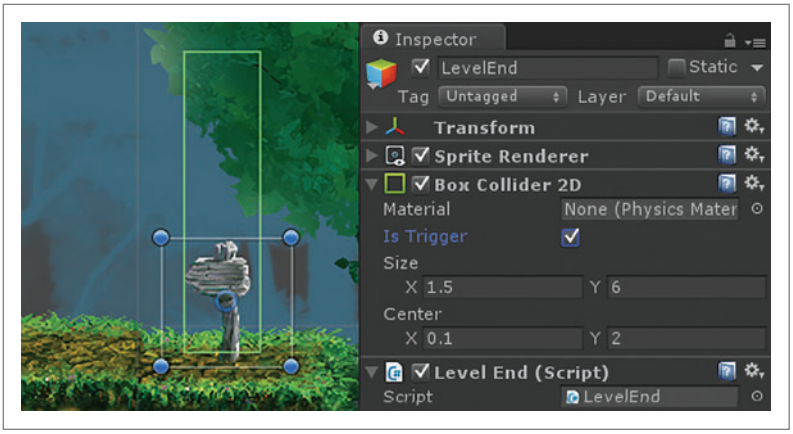


Figure 14 **The Game Object and Its Properties**

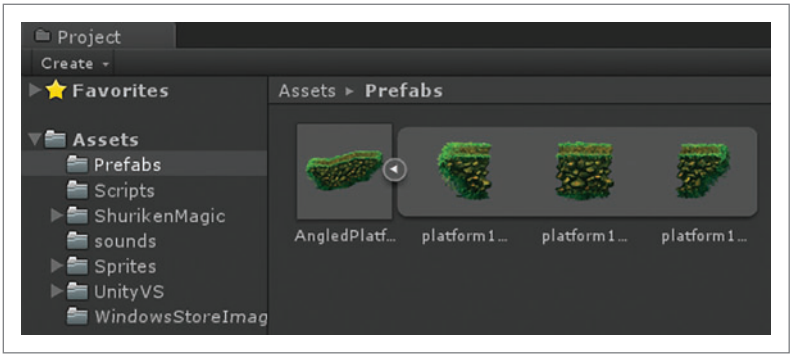


Figure 15 **Viewing the Contents of a Prefab**

allow the player to hit that zone. When it does, you just load another level or reload the current one:

```
void OnTriggerEnter2D(Collider2D collider)
{
  // If the player hits the trigger.
  if (collider.gameObject.tag == "Player")
  {
    // Reload the current level.
    Application.LoadLevel(Application.loadedLevel);

    // Could instead pass in the name of a scene to load:
    // Application.LoadLevel("Level2");
  }
}
```

The game object and its respective properties are shown in **Figure 14**. Note the collider's height is high enough that the player can't jump over it and also that this collider is set as a trigger.

Game Play In a simple 2D game like this, the flow is pretty straightforward. The player starts. Gravity on the rigid body makes the player fall. There's a collider on the player and on the platform, so the player stops. Keyboard, mouse, and touch input are read and moves the player. The player jumps between platforms by applying rigidbody.AddForce to make it jump, and moves left or right by reading Input.GetAxis-("Horizontal") and applying it to the rigidbody.velocity. The player picks up mushrooms, which are just colliders set as triggers. When the player touches them, they increment the score and destroy themselves. When the player finally makes it to the last sign, there's a collider/trigger that just reloads the current level. An additional to-do item here would be to add a large collider under the ground to detect when the player falls off the platform and simply reload the level then, as well.

Prefabs Reuse is important in coding, as well as in design. Once you assign several components and customize your game objects, you'll often want to reuse them across the same scene or even multiple scenes or games. You can create another instance of a game object in your scene, but you can also create an instance of a prefab that doesn't yet exist in your scene. Consider platforms and their colliders. If you want to reuse them across scenes, well, currently, you can't. But by creating prefabs, you can. Just drag any game object from the hierarchy back into the project folder and a new file is created with the extension .prefab that includes any child hierarchies. You can now drag this file into your scenes and reuse it. The original game object turns blue to note it's now connected to a prefab. Updating the .prefab file updates all instances in your scenes and you can also push changes from a modified scene prefab back down to the .prefab file, as well.

Clicking on the prefab displays the game objects it contains, as shown in **Figure 15**. If you make changes here, all instances in your scene will be updated.

Wrapping Up

There are a number of common operations performed across games. In this article, I covered the basics of a platformer game that uses colliders, rigid bodies, animations, score-keeping, basic GUI text and reading user input to apply force to move the player. These building blocks can be reused across a variety of game types. Stay tuned for a discussion of 3D coming in my next installment! ■

Adam Tuliper *is a senior technical evangelist with Microsoft living in sunny Southern California. He's an indie game dev, co-admin of the Orange County Unity Meetup, and a pluralsight.com author. He and his wife are about to have their third child, so reach out to him while he still has a spare moment at adamt@microsoft.com or on Twitter at twitter.com/AdamTuliper.*

Thanks to the following technical experts for reviewing this article: *Matt Newman (Subscience Studios), Tautvydas Žilys (Unity)*

ASPOSE.TOTAL

Powerful APIs which enable developers to harness the complexity of file format processing within their apps.

Your File Format APIs

Aspose.Words
DOC, DOCX, RTF, HTML, PDF, XPS & other document formats.

Aspose.Pdf
PDF, XML, XLS-FO, HTML, BMP, JPG, PNG & other image formats.

Aspose.Cells
XLS, XLSX, XLSM, XLTX, CSV, SpreadsheetML & image formats.

Aspose.Slides
PPT, PPTX, POT, POTX, XPS, HTML, PNG, PDF & other formats.

Aspose.Email
MSG, EML, PST, EMLX & other formats.

Aspose.BarCode
JPG, PNG, BMP, GIF, TIFF, WMF, ICON & other image formats.

Aspose.Imaging
PDF, BMP, JPG, GIF, TIFF, PNG, PSD & other image formats.

Aspose.Tasks
XML, MPP, SVG, PDF, TIFF, PNG, CSV, MPT & other formats.

Aspose.Diagram
VSD, VSDX, VSS, VST, VSX & other formats.

Aspose.Note
ONE, PNG, JPG, BMP, GIF & PDF.

... and more!

100% Standalone - No Office Automation

.NET Libraries | Java Libraries | Cloud APIs | Android Libraries

Scan for a 20% saving!

ASPOSE
Your File Format APIs

US: +1 888 277 6734
sales@aspose.com

EU: +44 141 416 1112
sales.europe@aspose.com

AU: +61 2 8003 5926
sales.asiapacific@aspose.com

GROUPDOCS.TOTAL

Professional APIs that allow developers to empower their apps with document collaboration capabilities.

Your Document Collaboration APIs

GroupDocs.Viewer

Native-text, high-fidelity HTML5 document viewer with support for over 49 file formats.

GroupDocs.Signature

Electronic signature API that gives your apps legally binding e-signature capabilities.

GroupDocs.Conversion

Universal document converter for fast conversion between more than 49 file formats.

GroupDocs.Annotation

A powerful API that lets developers annotate Microsoft Office, PDF and other documents within their own apps.

GroupDocs.Assembly

Incorporates data entered by users through online forms Into both Microsoft Office and PDF documents.

GroupDocs.Comparison

A diff view API that allows end users to quickly find differences between two revisions of a document.

100% Standalone - No Office Automation

.NET Libraries | Java Libraries | Cloud APIs | Cloud Apps

SALES INQUIRIES: +1 214 329 9760

sales@groupdocs.com

GROUPDOCS
Your Document Collaboration APIs

www.groupdocs.com

ASP.NET Web API Security Filters

Badrinarayanan Lakshmiraghavan

Authentication and authorization are cornerstones of application security. Authentication establishes the identity of a user by validating the credentials provided, while authorization determines whether a user is allowed to perform a requested action. A secured Web API authenticates requests and authorizes access to the resource requested based on the identity established.

You can implement authentication in ASP.NET Web API using the extensibility points available in the ASP.NET Web API pipeline, as well as using options provided by the host. With the first version of ASP.NET Web API, a common practice is to use an authorization filter or an action filter to implement authentication. ASP.NET Web API 2 introduces a new authentication filter dedicated to the process. This new extensibility point allows authentication and authorization concerns to be cleanly separated. In this article, I'm going to introduce you to these two security filters and show you how to use them to implement authentication and authorization as separate concerns in ASP.NET Web API.

Options for Implementing Security Aspects

Authentication and authorization in ASP.NET Web API can be implemented using the extensibility points offered by the host, as well as those available in the ASP.NET Web API pipeline itself. Host-based options include HTTP modules and OWIN middleware components, while ASP.NET Web API extensibility options consist of message handlers, action filters, authorization filters, and authentication filters.

This article discusses:

- Options for implementing authentication and authorization
- Getting authentication and authorization filters to work together
- Creating an authentication filter
- Using an authorization filter
- Filter overrides

Technologies discussed:

ASP.NET Web API, IIS, OWIN

Host-based options integrate well into the host pipeline and are capable of rejecting invalid requests earlier in the pipeline. The ASP.NET Web API extensibility options, on the other hand, offer a finer level of control over the authentication process. That is, you'll be able to set different authentication mechanisms for different controllers and even different action methods. The trade-off is better integration with the host and early rejection of bad requests versus the authentication granularity. Apart from these general characteristics, each option has its own pros and cons, as I'll cover in the sections that follow.

HTTP Module This is an option for Web APIs running on IIS. HTTP modules allow security code to execute early as part of the IIS pipeline. The principal established from an HTTP module is available to all components, including the IIS components running later in the pipeline. For example, when the principal is established by an HTTP module in response to the AuthenticateRequest event, the username of the principal gets logged correctly in the cs-username field in IIS logs. The biggest drawback with HTTP modules is the lack of granularity. HTTP modules run for all requests coming into the application. For a Web application with different capabilities such as HTML markup generation, Web APIs and so on, having an HTTP module enforcing authentication in one way is generally not a flexible-enough approach. Another disadvantage with using an HTTP module is the dependency on the host—IIS, in this case.

OWIN Middleware This is another host-related option, available with OWIN hosts. ASP.NET Web API 2 fully supports OWIN. Perhaps the most compelling reason for using OWIN middleware for security is that the same middleware can work across different frameworks. This means you can use multiple frameworks such as ASP.NET Web API, SignalR and so on in your application, yet use common security middleware. However, OWIN middleware's minimal granularity could be a shortcoming, because OWIN middleware runs in the OWIN pipeline and gets invoked typically for all requests. Also, OWIN middleware can be used only with OWIN-compatible hosts, although this dependency is comparatively better than taking dependency on a specific host/server such as IIS, as is the case with HTTP modules. A point worth noting here is

that OWIN middleware can run in the (IIS-integrated) ASP.NET pipeline, thanks to the package Microsoft.Owin.Host.SystemWeb.

Message Handler An extensibility option provided by ASP.NET Web API, the greatest benefit in using a message handler for security is it's a concept of the ASP.NET Web API framework and, hence, doesn't depend on the underlying host or server. Also, a message handler runs only for Web API requests. The downside of using a message handler is the lack of finer control. A message handler can be configured to run as a global handler for all requests or for a specific route. For a given route, you can have multiple controllers. All these controllers and the action methods they contain must share the same authentication enforced by the message handler configured for that route. In other words, the lowest granularity for authentication implemented by a message handler is at the route level.

Action Filter Another extensibility option provided by ASP.NET Web API is the action filter. However, from the perspective of implementing authentication, it isn't a viable option simply because it runs after the authorization filters are run in the ASP.NET Web API pipeline. For authentication and authorization to work correctly, authentication must precede authorization.

Authorization Filter Yet another extensibility option provided by ASP.NET Web API is the authorization filter. One of the most common ways of implementing custom authentication for scenarios requiring more granularity than message handlers can offer is to use an authorization filter. The major problem in using an authorization filter for both authentication and authorization is that the order of execution of authorization filters isn't guaranteed by ASP.NET Web API. Basically, this means your authorization filter performing authorization could very well run before your authorization filter performing authentication could run, making the authorization filter option equally unsuitable for authentication as the action filter option is.

Authentication Filter The focal point of this article, this is the newest extensibility option available with ASP.NET Web API 2. Authentication filters run after message handlers but before all other filter types. Therefore, they're a better choice for implementing authentication concerns. Most important, authentication filters run before authorization filters. By using a filter that specifically targets either authentication or authorization, you can separate authentication and authorization concerns.

Moreover, authentication filters offer a level of control or granularity that makes them particularly useful. Take the case of a Web API designed to be consumed by both native mobile applications and browser-based AJAX applications. The mobile app might present a token in the HTTP Authorization header while the AJAX app might use an authentication cookie as a credential. Further, suppose a subset of the API is sensitive and available only for the native mobile app and you want to ensure the action methods are accessed only by presenting a token and not a cookie (cookies are susceptible to cross-site request forgery [XSRF], while a token in an HTTP Authorization header is not). In this case, authentication must happen at a finer level of granularity than is possible with a host-based option or even a message handler. An authentication filter fits this use case perfectly. You can apply the authentication filter based on the token on all those controllers or action methods where they must be used and the authentication filter based on the cookie in other places. Suppose, in this scenario, you have a few common action methods and you want them accessible through either a token or a cookie. You can simply apply both the cookie and token authentication filters on those common action methods and one of the filters will be able to successfully authenticate. This kind of control is the biggest value authentication filters bring to the table. When granular control of authentication is required, the right approach is to implement authentication concerns through an authentication filter and authorization concerns through an authorization filter.

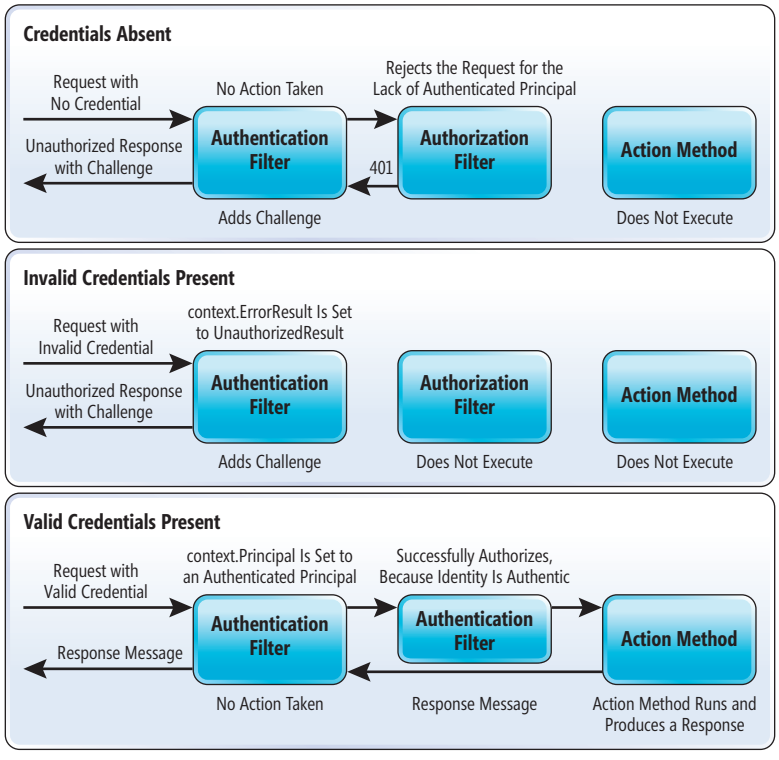


Figure 1 **Security Filters in ASP.NET Web API Pipeline**

It's worth mentioning here that the out-of-box authentication filter, HostAuthenticationFilter, enables ASP.NET Web API authentication via OWIN middleware. While OWIN authentication middleware runs in a pipeline and tries to "actively" authenticate the incoming requests, it can also be configured to authenticate the request "passively," only when asked. The HostAuthenticationFilter allows the running of passive OWIN authentication middleware by name later in the Web API pipeline. This approach enables authentication code that can be shared across frameworks (including the OWIN authentication middleware Microsoft provides) while still allowing per-action granularity for authentication.

Although you can mix host-level authentication with more-granular Web API pipeline-based authentication, you have to carefully consider how host-level authentication can affect Web API authentication. For example, you could have cookie-based authentication middleware at the host-level meant to be used with other frameworks, say ASP.NET MVC, but letting Web

API use the cookie-based principal makes it susceptible to attacks such as XSRF. To help in such situations, the SuppressDefaultHostAuthentication extension method enables Web API to ignore any authentication configured at the host level. The default Web API Visual Studio template has cookies enabled at the host level, and uses bearer tokens at the Web API level. Because cookies are enabled at the host level and require XSRF mitigation, the template also uses SuppressDefaultHostAuthentication to prevent the Web API pipeline from using the cookie-based principal. This way, Web API will use only the token-based principal and you don't need to build mechanisms for Web API to defend against XSRF attacks.

Making Authentication and Authorization Filters Work in Tandem

In the ASP.NET Web API pipeline, authentication filters run first—followed by the authorization filters— for the simple reason that authorization depends on the identity established, which is the outcome of authentication. Here's how you can design the authentication and authorization filters to work together to secure ASP.NET Web API.

The basic tenet of this design is giving the authentication filter the one and only responsibility of validating the credential, and not having it deal with any other concerns. For example, the authentication filter will not reject a request with the 401 Unauthorized status code if credentials aren't present. It simply doesn't establish an authenticated identity, and leaves the question of how to handle anonymous requests to the authorization stage. An authentication filter basically takes three types of actions:

1. If the credential of interest isn't present in the request, the filter does nothing.
2. If credentials are present and found to be valid, the filter establishes an identity in the form of an authenticated principal.
3. If credentials are present but found to be invalid, the filter notifies the ASP.NET Web API framework by setting an error result, which basically results in an "unauthorized" response that gets sent back to the requestor.

If none of the authentication filters running in the pipeline can detect an invalid credential, the pipeline continues to run, even if an authenticated identity hasn't been established. It's up to the components running later in the pipeline to decide how to handle this anonymous request.

At the most fundamental level, an authorization filter simply checks whether the established identity is an authenticated identity. However, an authorization filter can also ensure that:

- The username of the authenticated identity is on the list of allowed users.
- At least one of the roles associated with the authenticated identity is on the list of allowed roles.

While the out-of-box authorization filter performs only the role-based access control as just described, a custom authorization filter that derives from the out-of-box authorization filter can perform claims-based access control by checking the claims that are part of the identity established by the authentication filter.

If all the authorization filters are happy, the pipeline continues to execute and eventually the action method of the API controller generates a response for the request. If the identity isn't established or if there's a mismatch in terms of the username or the role requirements, the authorization filter rejects the request with a 401 Unauthorized response. **Figure 1** illustrates the part played by the two filters in three scenarios: credentials not present, invalid credentials present and valid credentials present.

Creating an Authentication Filter

An authentication filter is a class implementing the IAuthenticationFilter interface. This interface has two methods: AuthenticateAsync and ChallengeAsync, as shown by the following:

```
public interface IAuthenticationFilter : IFilter
{
  Task AuthenticateAsync(HttpAuthenticationContext context,
    CancellationToken cancellationToken);
  Task ChallengeAsync(HttpAuthenticationChallengeContext context,
    CancellationToken cancellationToken);
}
```

The AuthenticateAsync method accepts HttpAuthenticationContext as an argument. This context is how the AuthenticateAsync method communicates the result of the authentication process back to the ASP.NET Web API framework. If the request message contains authentic credentials, the Principal property of the passed-in HttpAuthenticationContext object is set to the authenticated principal. If the credentials are invalid, the ErrorResult property of the HttpAuthenticationContext parameter is set to UnauthorizedResult. If the request message doesn't contain the credential at all, the AuthenticateAsync method takes no action. The code in **Figure 2** shows a typical implementation of the AuthenticateAsync method covering these three scenarios. The authenticated principal used in this example is a ClaimsPrincipal with just name and role claims.

You use the AuthenticateAsync method to implement the core authentication logic of validating the credentials in the request, and

Figure 2 **The AuthenticateAsync Method**

```
public Task AuthenticateAsync(HttpAuthenticationContext context,
  CancellationToken cancellationToken)
{
  var req = context.Request;
  // Get credential from the Authorization header (if present) and authenticate
  if (req.Headers.Authorization != null &&
    "somescheme".Equals(req.Headers.Authorization.Scheme,
      StringComparison.OrdinalIgnoreCase))
  {
    var creds = req.Headers.Authorization.Parameter;
    if(creds == "opensesame") // Replace with a real check
    {
      var claims = new List<Claim>()
      {
        new Claim(ClaimTypes.Name, "badri"),
        new Claim(ClaimTypes.Role, "admin")
      };
      var id = new ClaimsIdentity(claims, "Token");
      var principal = new ClaimsPrincipal(new[] { id });

      // The request message contains valid credential
      context.Principal = principal;
    }
    else
    {
      // The request message contains invalid credential
      context.ErrorResult = new UnauthorizedResult(
        new AuthenticationHeaderValue[0], context.Request);
    }
  }

  return Task.FromResult(0);
}
```

GdPicture.NET 10

- Document viewing, processing, printing and scanning (TWAIN & WIA).
- Reading, writing and converting vector and raster images in more than 90 formats, PDF included.
- OMR, OCR, barcode reading and writing (linear & 2D).
- Annotations for image and PDF within Windows and Web applications.
- Color detection engine for image and PDF compression.

And much more ...

All-In-One Document Imaging SDK

Royalty-Free Document Imaging Toolkits for .NET and COM/ActiveX

GdPicture.NET 10 Plugins

- NEW Color detection
- NEW DICOM image reader
- NEW MICR reader
- Full managed PDF support
- Full annotations support for PDF and images
- OCR
- Forms processing
- JBIG2 encoding
- 1D and 2D barcode reading and writing

Try GdPicture.NET 10 FREE for 30 days

www.gdpicture.com

the ChallengeAsync method to add the authentication challenge. An authentication challenge is added to a response when the status code is 401 Unauthorized, and to inspect the status code, you need the response object. But the ChallengeAsync method doesn't allow you to inspect the response or set the challenge directly. In fact, this method executes before the action method, in the request processing part of the Web API pipeline. However, the parameter of the ChallengeAsync method HttpAuthenticationChallengeContext allows an action result object (IHttpActionResult) to be assigned to the Result property. The ExecuteAsync method of the action result object awaits the task producing the response, inspects the response status code, and adds the WWW-Authenticate response header. The code in **Figure 3** shows a typical implementation of the ChallengeAsync method. In this example, I just add a hardcoded challenge. The ResultWithChallenge class is the action result class I created to add the challenge.

The following code shows the completed filter class:

```
public class TokenAuthenticationAttribute : Attribute, IAuthenticationFilter
{
  public bool AllowMultiple { get { return false; } }

  // The AuthenticateAsync and ChallengeAsync methods go here
}
```

In addition to implementing the IAuthenticationFilter interface, I derive from Attribute so this class can be applied as an attribute at the class (controller) level or method (action method) level.

Thus, you can create an authentication filter with the single responsibility of authenticating a specific credential (a fictitious token in this example). An authentication filter has no authorization logic; its only purpose is to handle authentication: establishing an identity, if any, (while processing the request message) and returning a challenge, if any (while processing the response message). Authorization filters handle authorization concerns, such as checking whether the identity is an authenticated identity or whether it's on the allowed list of users or roles.

Figure 3 **The ChallengeAsync Method**

```
public Task ChallengeAsync(HttpAuthenticationChallengeContext context,
  CancellationToken cancellationToken)
{
  context.Result = new ResultWithChallenge(context.Result);
  return Task.FromResult(0);
}

public class ResultWithChallenge : IHttpActionResult
{
  private readonly IHttpActionResult next;

  public ResultWithChallenge(IHttpActionResult next)
  {
    this.next = next;
  }

  public async Task<HttpResponseMessage> ExecuteAsync(
    CancellationToken cancellationToken)
  {
    var response = await next.ExecuteAsync(cancellationToken);

    if (response.StatusCode == HttpStatusCode.Unauthorized)
    {
      response.Headers.WwwAuthenticate.Add(
        new AuthenticationHeaderValue("somescheme", "somechallenge"));
    }
    return response;
  }
}
```

Using an Authorization Filter

The fundamental goal of using an authorization filter is to perform authorization—to determine whether a user should be allowed access to a requested resource. Web API provides an implementation of an authorization filter called AuthorizeAttribute. Applying this filter ensures the identity is an authenticated identity. You can also configure the authorize attribute with a list of specific usernames and the roles to allow. The code in **Figure 4** shows the authorization filter applied at different levels (globally, at the controller level and at the action method level), using different attributes of the identity to authorize. The filter in this example globally ensures the identity is authenticated. The filter used at the controller level ensures the identity is authenticated and that at least one role associated with the identity is "admin." The filter used at the action method level ensures the identity is authenticated and that the username is "badri." One point to note here is that the authorization filter at the action method level also inherits the filters from the controller and global levels. Hence, for authorization to succeed all filters must pass: the username must be "badri," one of the roles must be "admin" and the user must be authenticated.

The out-of-box AuthorizeAttribute is extremely useful, but if more customization is desired, you can sub-class it to implement additional authorization behavior. The following code shows a custom authorization filter:

```
public class RequireAdminClaimAttribute : AuthorizeAttribute
{
  protected override bool IsAuthorized(HttpActionContext context)
  {
    var principal =
      context.Request.GetRequestContext().Principal as ClaimsPrincipal;

    return principal.Claims.Any(c => c.Type ==
      "http://yourschema/identity/claims/admin"
      && c.Value == "true");
  }
}
```

This filter simply checks for an "admin" custom claim, but you can use a principal and any other additional information from the HttpActionContext to perform custom authorization here.

In the first version of ASP.NET Web API, custom authorization filters were often misused to implement authentication, but with ASP.NET Web API 2, authentication filters now have their own place in the pipeline and this helps the development of clean, modular code with authentication and authorization concerns clearly separated.

Filter Overrides

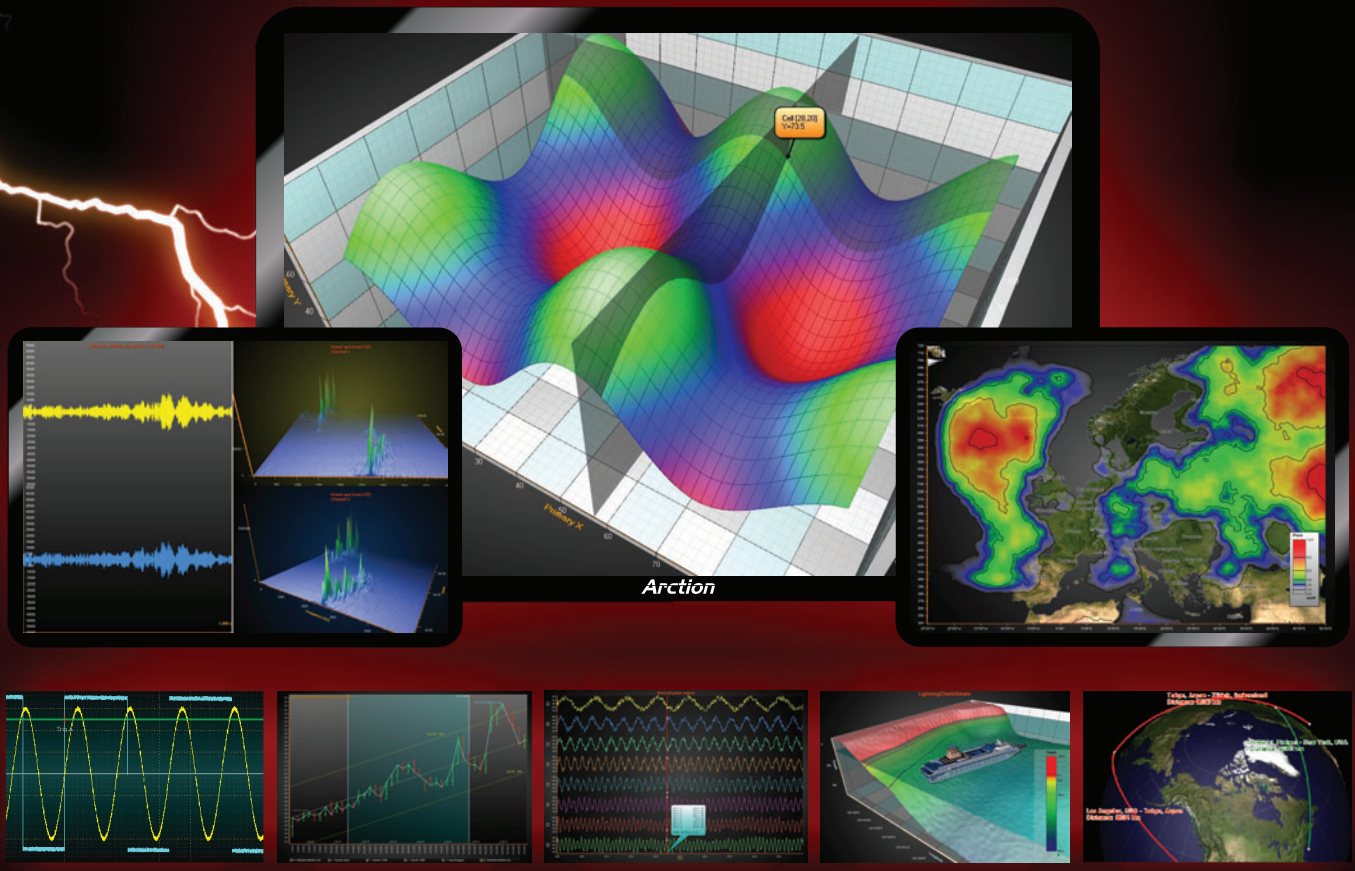
As I explained earlier, an authorization filter can be applied at the action method level, the controller level or globally. By specifying the Authorize filter globally, you can enforce authorization on all action method invocations across all controllers. If you want a few methods to be exempted from the globally configured check, it's easier to accomplish this using the AllowAnonymous attribute.

The code in **Figure 5** shows the use of the AllowAnonymous attribute at the controller level. Though the authorization filter is applied globally, the AllowAnonymous attribute used with PublicResourcesController exempts the requests coming to this controller from getting authorized.

AllowAnonymous provides a way a specific action can override authorization configured by a higher-level authorization filter.

The fastest rendering data visualization components
for WPF and WinForms...

LightningChart



HEAVY-DUTY DATA VISUALIZATION TOOLS FOR SCIENCE, ENGINEERING AND TRADING

WPF charts performance comparison

Opening large dataset	LightningChart is up to 977,000 % faster
Real-time monitoring	LightningChart is up to 2,700,000 % faster

Winforms charts performance comparison

Opening large dataset	LightningChart is up to 37,000 % faster
Real-time monitoring	LightningChart is up to 2,300,000 % faster

Results compared to average of other chart controls. See details at www.LightningChart.com/benchmark. LightningChart results apply for Ultimate edition.

- Entirely DirectX GPU accelerated
- Superior 2D and 3D rendering performance
- Optimized for real-time data monitoring
- Touch-enabled operations
- Supports gigantic data sets
- On-line and off-line maps
- Great customer support
- Compatible with Visual Studio 2005...2013

Download a free 30-day evaluation from

www.LightningChart.com

Arction

Pioneers of high-performance data visualization

However, AllowAnonymous only allows you to override authorization. Suppose you want most actions to authenticate using HTTP Basic authentication, but one action to authenticate using just tokens. It would be nice to configure token authentication globally but then override authentication for this one action, similarly to how AllowAnonymous overrides authorization.

ASP.NET Web API 2 introduces a new filter type to address this scenario, the override filter. Unlike AllowAnonymous, the override filters introduced in ASP.NET Web API 2 can work with any type of filter. Override filters, as the name suggests, override the filters configured at higher levels. To override the authentication filters configured at higher levels, use the out-of-box attribute OverrideAuthentication. If you have an authentication filter applied globally and want to stop it from running for a specific action method or controller, you can simply apply OverrideAuthentication at the desired level.

Override filters are useful for far more than simply stopping certain filters from running. Say you have two authentication filters, one for authenticating a security token and another for authenticating a username/password in an HTTP basic scheme. Both filters are applied globally, making your API flexible enough to accept either the token or username/password. The following code shows the two authentication filters applied globally:

```
public static class WebApiConfig
{
  public static void Register(HttpConfiguration config)
  {
    // Other Web API configuration code goes here
    config.Filters.Add(new TokenAuthenticator());
    config.Filters.Add(new HttpBasicAuthenticator(realm: "Magical"));
  }
}
```

Now, perhaps you'd like to ensure only the token is used as a credential to access a specific action method. How will OverrideAuthentication enable you to meet this need, given it just stops all filters from running? Here's the important characteristic of the override filters—they wipe out all filters specified at higher levels but don't remove filters specified at the same level as themselves. This basically means you can add one or more authentication filter at a specific level while eliminating everything else at higher levels. Going back to the requirement of using just the token as a credential for accessing a specific action method, you can simply specify the OverrideAuthentication attribute and TokenAuthenticator at the action method level, as shown in the following code (this ensures only TokenAuthenticator is run for the action method GetAllowedForTokenOnly):

```
public class EmployeesController : ApiController
{
  [OverrideAuthentication] // Removes all authentication filters
  [TokenAuthenticator] // Puts back only the token authenticator
  public string GetAllowedForTokenOnly(int id)
  {
    return "Hello World";
  }
}
```

Thus, the override filters introduced with ASP.NET Web API 2 provide a lot more flexibility in terms of specifying filters globally and running filters at lower levels selectively only where the global behavior has to be overridden.

In addition to the OverrideAuthentication attribute, there's also an out-of-box attribute called OverrideAuthorization that removes authorization filters specified at higher levels. Compared with AllowAnonymous, the difference is that OverrideAuthorization removes only the higher-level authorization filters. It doesn't remove authorization filters specified at the same level as itself. AllowAnonymous makes ASP.NET Web API skip the authorization process all together—even if there are authorization filters specified at the same level as AllowAnonymous, they're simply ignored.

Figure 4 **Using an Authorization Filter at Three Different Levels**

```
public static class WebApiConfig
{
  public static void Register(HttpConfiguration config)
  {
    // Other Web API configuration code goes here
    config.Filters.Add(new AuthorizeAttribute()); // Global level
  }
}

[Authorize(Roles="admin")] // Controller level
public class EmployeesController : ApiController
{
  [Authorize(Users="badri")] // Action method level
  public string Get(int id)
  {
    return "Hello World";
  }
}
```

Figure 5 **Using the AllowAnonymous Attribute**

```
public static class WebApiConfig
{
  public static void Register(HttpConfiguration config)
  {
    // Other Web API configuration code goes here
    config.Filters.Add(new AuthorizeAttribute()); // Global level
  }
}

[AllowAnonymous]
public class PublicResourcesController : ApiController
{
  public string Get(int id)
  {
    return "Hello World";
  }
}
```

Wrapping Up

You can implement authentication in ASP.NET Web API using the options provided by the host, as well as the extensibility points provided by the ASP.NET Web API pipeline. Host-based options integrate well into the host pipeline and reject invalid requests earlier in the pipeline. ASP.NET Web API extensibility points offer a finer level of control over the authentication process. When you require more control of authentication, for example, you want to use different authentication mechanisms for different controllers or even different action methods, the right approach is to implement authentication concerns through an authentication filter and authorization concerns through an authorization filter. ■

Badrinarayanan Lakshmiraghavan *is a senior consulting architect with a Fortune 500 company. He's the author of the books, "Pro ASP.NET Web API Security" and "Practical ASP.NET Web API" both published by Apress in 2013. He occasionally blogs at lbadri.wordpress.com.*

Thanks to the following Microsoft technical expert for reviewing this article: *David Matson*

JAMES MCCAFFREY

Winnow Classification Using C#

In machine learning (ML), a classification problem is one in which the goal is to create a model that predicts a result that takes on discrete, non-numeric values. For example, you might want to predict the political party (Democrat or Republican) of a person. There are many different classification algorithms and techniques, including, for example, naive Bayes classification, logistic regression classification, and neural network classification.

A very simple and interesting classification technique is using the Winnow algorithm. (The English word *winnow* means to remove unwanted items). The best way to get a feel for what Winnow classification is and to see where this article is headed is to take a look at the demo program in **Figure 1**.

The goal of the demo program is to predict the political party, Democrat or Republican, of a member of the U.S. House of Representatives, based on the Representative's votes on 16 bills. The House of Representatives has 435 members. A well-known benchmark data set contains 435 items stored in a simple text file. The first three data items are:

```
republican,n,y,n,y,y,y,n,n,n,y,?,y,y,y,n,y
republican,n,y,n,y,y,y,n,n,n,n,n,y,y,y,n,?
democrat,?,y,y,?,y,y,n,n,n,n,y,n,y,y,n,n
```

The first column in a line is either democrat or republican. The next 16 columns are either n (a no vote), y (a yes vote), or ? (missing, unknown or abstain). The first column represents the member's vote on a bill related to handicapped infants. The second is a bill related to a water project. And so on. You don't need to know what bill is associated with each column, but if you're interested you can find this data set (with descriptions) in many places on the Internet by doing a search for "UCI voting data."

Winnow classification is designed for a very specific type of classification problem: one where the class to predict can take one of two possible values (these are called binary classification problems), and the predictor variables (often called "features" in ML terminology) also can take one of two possible values. The raw voting data set meets these requirements if the missing values are dealt with. The most common approach for dealing with missing values is simply to delete any data item that has one or more missing values. However, in the case of voting data, a missing value often represents an implied "no" vote, so the demo program replaces all missing values with "no" votes.

The demo program uses the first 100 data items in the voting set rather than all 435 items. The demo encodes independent variable "no" votes as 0 and "yes" votes as 1, and encodes the dependent political party X variable values "democrat" as 0 and "republican"

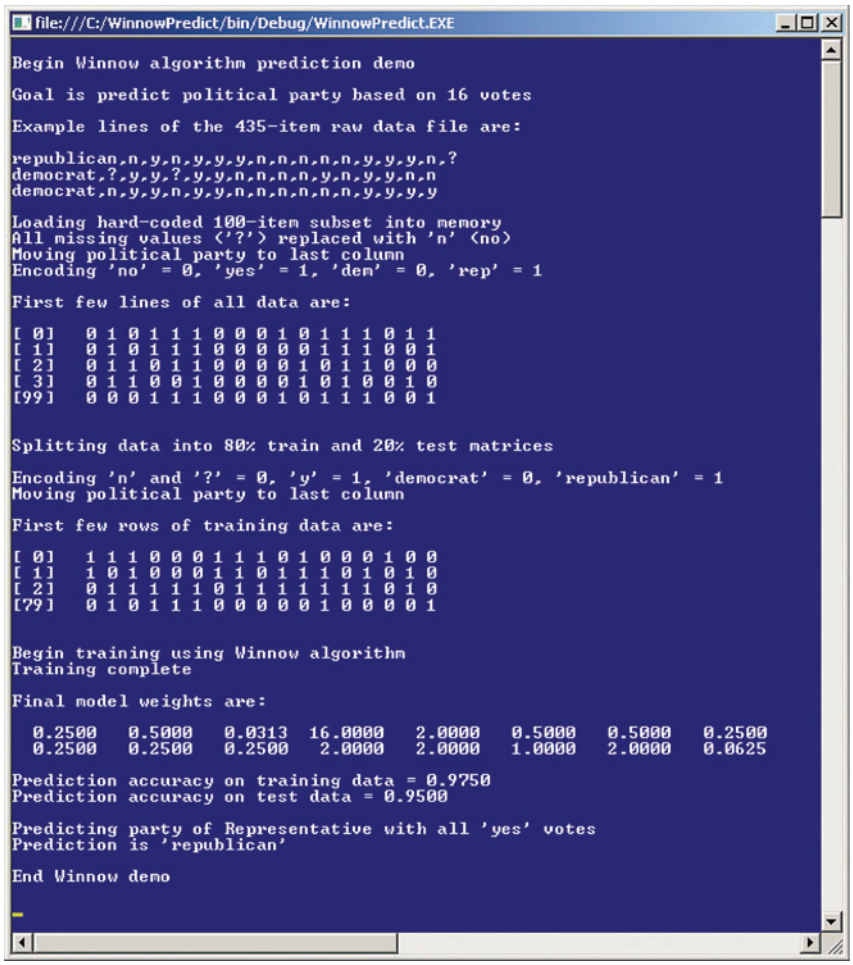


Figure 1 **Winnow Algorithm Classification Demo**

Code download available at msdn.microsoft.com/magazine/msdnmag0914.

as 1. The demo moves the dependent Y variable from the first column to the last column because using the last column for Y is more convenient when coding, and more common.

The demo randomly splits the 100-item encoded data set into an 80-item training set to generate the model, and a 20-item test set to evaluate the accuracy of the model. The demo uses the Winnow algorithm to find 16 weights (one for each input): 0.2500, 0.500, . . , 0.0625. Using these weights, the demo model correctly predicts the political party of 97.50 percent of the training items (78 out of 80) and 95.00 percent of the test items (19 out of 20).

A very simple and interesting classification technique is using the Winnow algorithm.

The demo concludes by using the model to predict the political party of a hypothetical Representative who voted "yes" on all 16 bills. Based on the model, the prediction is that the person is a Republican.

This article assumes you have at least intermediate programming skills but doesn't assume you know anything about the Winnow classification algorithm. The demo is coded using C#, but you shouldn't have much trouble if you want to refactor the code to another language, such as Visual Basic .NET or JavaScript.

Understanding the Winnow Algorithm

For simplicity, suppose the training data is based on just five votes rather than 16. For example:

```
1, 0, 1, 1, 0 -> 0
1, 1, 1, 0, 1 -> 1
0, 0, 0, 1, 0 -> 1
```

So, the first line means "yes-vote," "no-vote," "yes-vote," "yes-vote," "no-vote," "democrat." Now suppose that a model is trained and yields weights { 0.75, 3.00, 1.00, 0.50, 0.40 }. To compute a predicted output, the algorithm just multiplies each input value by the corresponding weight, sums the terms and then compares the accumulated sum against a threshold value. For example, for the first item, the accumulated sum is:

```
(1 * 0.75) + (0 * 3.00) + (1 * 1.00) + (1 * 0.50) + (0 * 0.40) = 2.25
```

If the threshold value is 5.0 then, because the accumulated sum (2.25) is less than the threshold, the predicted Y is 0, which in this case is a correct prediction.

Training a model involves finding a set of weights so that when applied to training data with known output values, the computed outputs closely match the known outputs. In high-level pseudo-code the Winnow algorithm is:

```
initialize weights
loop until done
  for each training item
    compute Y
    if correct do nothing
    else if incorrect
      if computed Y is too large
        divide all relevant weights by 2.0
      else if computed Y is too small
        multiply all relevant weights by 2.0
  end for
end loop
```

Suppose for the three data items mentioned earlier, weights are initialized to:

```
{ 2.50, 2.50, 2.50, 2.50, 2.50 }
```

The computed Y for the first training item (1, 0, 1, 1, 0 -> 0) is:

```
Y = (1 * 2.50) + (0 * 2.50) + (1 * 2.50) + (1 * 2.50) + (0 * 2.50) = 7.50 -> 1
```

This is an incorrect prediction where the computed Y of 1 is too big compared to the desired Y of 0, so the relevant weights are divided by a factor of 2.0. The relevant weights are those weights associated with 1 inputs. The new weights are:

```
{ 2.50 / 2, (no change), 2.50 / 2, 2.50 / 2, (no change) }
= { 1.25, 2.50, 1.25, 1.25, 2.50 }
```

Now, the second training item (1, 1, 1, 0, 1 -> 1) is processed using the new weights:

```
Y = (1 * 1.25) + (1 * 2.50) + (1 * 1.25) + (0 * 1.25) + (1 * 2.50) = 7.50 -> 1
```

This is a correct prediction, so all weights are left alone. The third training item (0, 0, 0, 1, 0 -> 1) is processed using the current weights:

```
Y = (0 * 1.25) + (0 * 2.50) + (0 * 1.25) + (1 * 1.25) + (0 * 2.50) = 1.25 -> 0
```

This is an incorrect prediction where the computed Y of 0 is too small compared to the desired Y of 1, so relevant weights are multiplied by 2.0:

```
{ (no change), (no change), (no change), 1.25 * 2.0, (no change) }
= { 1.25, 2.50, 1.25, 2.50, 2.50 }
```

Figure 2 **Overall Demo Program Structure**

```
using System;
namespace WinnowPredict
{
  class WinnowProgram
  {
    static void Main(string[] args)
    {
      Console.WriteLine("Begin Winnow algorithm demo");

      int[][] data = new int[100][];
      data[0] = new int[] { 0, 1, 0, 1, 1, 1, 0, 0,
        0, 1, 0, 1, 1, 1, 0, 1, 1 };
      // Etc.

      // Create, train and test matrices from data
      // Instantiate Winnow classifier
      // Train using training data
      // Compute accuracy
      // Predict party of hypothetical person

      Console.WriteLine("End Winnow demo");
      Console.ReadLine();
    } // Main

    static void MakeTrainTest(int[][] data, int seed,
      out int[][] trainData, out int[][] testData) { . . }
    static void ShowVector(double[] vector, int decimals,
      int valsPerRow, bool newLine) { . . }
    static void ShowMatrix(int[][] matrix, int numRows,
      bool indices) { . . }
  }

  public class Winnow
  {
    private int numInput;
    private double[] weights;
    private double threshold;
    private double alpha;
    private static Random rnd;

    public Winnow(int numInput, int rndSeed) { . . }
    public int ComputeOutput(int[] xValues) { . . }
    public double[] Train(int[][] trainData) { . . }
    public double Accuarcy(int[][] trainData) { . . }
    private static void Shuffle(int[][] trainData) { . . }
  }
}
```

Figure 3 **The Main Method Setting up a 100-Item Data Set**

```
using System;
namespace WinnowPredict
{
  class WinnowProgram
  {
    static void Main(string[] args)
    {
      Console.WriteLine("Begin Winnow algorithm prediction demo");

      int[][] data = new int[100][];
      data[0] = new int[] { 0, 1, 0, 1, 1, 1, 0, 0,
        0, 1, 0, 1, 1, 1, 0, 1, 1 };
...
```

If you think about the Winnow training process a bit, you'll notice two key characteristics. First, the algorithm uses only incorrect information. Second, the algorithm essentially drives the weights of irrelevant predictors to 0, winnowing them out. This often makes Winnow classification extremely effective in situations where many of the predictor variables are irrelevant. Although the basic Winnow algorithm is quite simple, there are several implementation details to solve, such as deciding how to initialize the weights and determining when to stop training.

Overall Program Structure

The overall structure of the demo program, with a few WriteLine statements removed and minor edits to save space, is presented in **Figure 2**. To create the demo program, I launched Visual Studio and created a new C# console application named WinnowPredict. The demo has no significant .NET version dependencies, so any version of Visual Studio should work. After the template code loaded into the editor, I deleted all using statements except for the single reference to the top-level System namespace. In the Solution Explorer window, I renamed file Program.cs to WinnowProgram.cs and Visual Studio automatically renamed class Program for me.

The demo has no significant .NET version dependencies, so any version of Visual Studio should work.

The demo program is a bit too long to present in its entirety in this article, but you can find the complete source code (WinnowProgram.cs) in the accompanying file download. All the Winnow classification logic is contained in a program-defined class named Winnow.

The Main method begins by setting up a 100-item data set, as shown in **Figure 3**.

To create the data, I located the UCI voting data set on the Web, copied and pasted it into Notepad, and then used edit-replace, placing the data directly into an array-of-arrays-style matrix. Even if you're an experienced programmer, you might not be familiar with C# matrix syntax, but you'll likely get used to the coding patterns quickly. In a non-demo scenario, you'd write some sort of method to load the data from the text file into a matrix.

Next, the first four lines and last line of the entire data set are displayed using a helper method, and the data set is split into a training set (80 items) and test set (20 items):

```
Console.WriteLine("\nFirst few lines of all data are: \n");
ShowMatrix(data, 4, true);
Console.WriteLine("\nSplitting data into 80% train" +
  " and 20% test matrices");
int[][] trainData = null;
int[][] testData = null;
MakeTrainTest(data, 0, out trainData, out testData);
```

Method MakeTrainTest has a hardcoded 80 percent to 20 percent split (you might want to parameterize the percent to use as training data). After displaying part of the training data to verify nothing went wrong, the demo instantiates a Winnow classifier object and uses the Train method to create a model, that is, to find a set of good weights:

```
Console.WriteLine("First few rows of training data are:");
ShowMatrix(trainData, 3, true);

Console.WriteLine("Begin training using Winnow algorithm");
int numInput = 16;
Winnow w = new Winnow(numInput, 0);
weights = w.Train(trainData);
Console.WriteLine("Training complete");
```

The Winnow constructor requires the number of x-variables and a seed value for random number generation, which is used to process the training items in random order. Next, the demo displays the final weights found by the Train method, and then computes the model's accuracy on the training and test sets:

```
Console.WriteLine("Final model weights are:");
ShowVector(weights, 4, 8, true);
double trainAcc = w.Accuarcy(trainData);
double testAcc = w.Accuarcy(testData);
Console.WriteLine("Prediction accuracy on training data = " +
  trainAcc.ToString("F4"));
Console.WriteLine("Prediction accuracy on test data = " +
  testAcc.ToString("F4"));
```

The demo concludes by predicting the political party of a hypothetical U.S. House Representative who voted "yes" on all 16 bills, as shown in **Figure 4**.

The Winnow Class

The Winnow class has five data members:

```
private int numInput;
private double[] weights;
private double threshold;
private double alpha;
private static Random rnd;
```

Data member threshold holds the value used to determine if computed Y is 0 (below threshold) or 1 (above threshold). Data

Figure 4 **Predicting the Political Party**

```
...
  Console.WriteLine("Predicting party of Representative" + "
    with all 'yes' votes");
  int[] unknown = new int[] { 1, 1, 1, 1, 1, 1, 1, 1,
  1 , 1, 1, 1, 1, 1, 1, 1 };
  int predicted = w.ComputeOutput(unknown);
  if (predicted == 0)
    Console.WriteLine("Prediction is 'democrat'");
  else
    Console.WriteLine("Prediction is 'republican'");

  Console.WriteLine("End Winnow demo");
  Console.ReadLine();
} // Main
```

Figure 5 **Updating Weights**

```
if (computed == 1 && target == 0) // Decrease
{
  for (int j = 0; j < numInput; ++j)
  {
    if (xValues[j] == 0) continue; // Not relevant
    weights[j] = weights[j] / alpha; // Demotion
  }
}
else if (computed == 0 && target == 1) // Increase
{
  for (int j = 0; j < numInput; ++j)
  {
    if (xValues[j] == 0) continue; // Not relevant
    weights[j] = weights[j] * alpha; // Promotion
  }
}
```

member alpha is the factor to decrease (called demotion in research literature) or increase (called promotion) weights when an incorrect Y is computed. The standard value for alpha is 2.0, as shown earlier in this article. You may want to experiment by using two different values for promotion and demotion, instead of a single alpha value for both.

The Winnow constructor is defined like so:

```
public Winnow(int numInput, int rndSeed)
{
  this.numInput = numInput;
  this.weights = new double[numInput];
  for (int i = 0; i < weights.Length; ++i)
    weights[i] = numInput / 2.0;
  this.threshold = 1.0 * numInput;
  this.alpha = 2.0;
  rnd = new Random(rndSeed);
}
```

The threshold value is initialized to the number of input variables, for example 16.0 in the demo. This value is standard for the Winnow algorithm, but you might want to experiment with alternative values. The no-effect multiplication by 1.0 is there to suggest that different values can be used, but they should be related to the number of input variables.

The weights array is allocated using the number of input variables, 16 in the demo. Each of the weights is initialized to the number of input variables divided by 2, or 8.0 in the demo. This value is arbitrary and, again, you might want to experiment. The Winnow algorithm hasn't been explored by research very much relative to many other, more commonly used classification algorithms.

Method ComputeOutput is straightforward:

```
public int ComputeOutput(int[] xValues)
{
  double sum = 0.0;
  for (int i = 0; i < numInput; ++i)
    sum += weights[i] * xValues[i];
  if (sum > this.threshold)
    return 1;
  else
    return 0;
}
```

Notice that a 1 value is returned if the accumulated sum is strictly greater than the threshold value, as opposed to greater than or equal to the threshold. This is arbitrary and using ">=" instead of ">" will not have a significant impact on the algorithm.

The Training Method

The Winnow algorithm training routine is one of the shortest and simplest of all ML classification algorithms. The method definition begins:

```
public double[] Train(int[][] trainData)
{
  int[] xValues = new int[numInput];
  int target;
  int computed;
  Shuffle(trainData);
...
```

Local array "xValues" holds the input values from a training item, but not the target output value. Local variable "target" holds the desired Y value from a training data item. Local variable "computed" stores the computed Y value, for a given set of input x-values and the current set of weights. Method Shuffle randomizes the order of the training data using the Fisher-Yates mini-algorithm.

Next, the main training loop begins:

```
for (int i = 0; i < trainData.Length; ++i)
{
  Array.Copy(trainData[i], xValues, numInput); // Inputs
  target = trainData[i][numInput]; // Last value is target
  computed = ComputeOutput(xValues);
  // Update weights here
}
```

As mentioned earlier, it's not entirely clear how long to train. The demo iterates just once through the training data. An alternative is to iterate multiple times through the training data, stopping after a fixed number of iterations, or perhaps when some desired accuracy has been reached.

Inside the main training loop, weights are updated using the code in **Figure 5**.

Method Train concludes by copying the best weights found into a local return-array:

```
...
  } // end for
  double[] result = new double[numInput]; // = number weights
  Array.Copy(this.weights, result, numInput);
  return result;
} // Train
```

Wrapping Up

This article is based on the research paper that originally presented the Winnow algorithm, "Learning Quickly When Irrelevant Attributes Abound: A New Linear-Threshold Algorithm" (1998), N. Littlestone. You can find this paper in PDF format on the Web in several locations.

While doing background research for this article, I came across an interesting implementation of the Winnow algorithm written using the F# language. The implementation is in a personal blog post written by Mathias Brandewinder (bit.ly/1z8hfj6).

Although Winnow classification is designed specifically for binary classification problems where the independent x-variables are all binary, you might want to experiment on problems where one or more x-variables are categorical. For example, suppose one predictor variable is "color" and can take one of three possible values: "red," "green," or "blue." If you use 1-of-N encoding, red becomes (1, 0, 0), green becomes (0, 1, 0), blue becomes (0, 0, 1), and the Winnow algorithm can be applied. ■

Dr. James McCaffrey *works for Microsoft Research in Redmond, Wash. He has worked on several Microsoft products including Internet Explorer and Bing. He can be reached at jammc@microsoft.com.*

Thanks to the following technical experts at Microsoft Research for reviewing this article: *Nathan Brown and Kirk Olynyk*

Visual Studio LIVE!

EXPERT SOLUTIONS FOR .NET DEVELOPERS

WASHINGTON, D.C.

October 6 – 9, 2014

Washington Marriott at Metro Center

YOUR GUIDE TO THE .NET DEVELOPMENT UNIVERSE

Register by September 10 and Save $200!

Use promo code DCSEP2

Scan the QR code to register or for more event details.

CONNECT WITH VISUAL STUDIO LIVE!

twitter.com/vslive – @VSLive

facebook.com – Search "VSLive"

linkedin.com – Join the "Visual Studio Live" group!

vslive.com/dc

THAT'S RIGHT, WE'RE TRANSPORTING Visual Studio Live! to our nation's capital for the first time in 21 years. From Oct 6 – 9, 2014, developers, software architects, engineers and designers will gather for 4 days of cutting-edge education on the Microsoft Platform.

Explore HOT TOPICS like:

- Visual Studio 2013 IDE
- MVC 5
- Breeze
- C# 6.0
- Angular
- Entity Framework
- Xamarin
- And Many More!

SUPPORTED BY

Microsoft Visual Studio msdn magazine Visual Studio MAGAZINE

PRODUCED BY

1105 MEDIA

Washington, D.C. AGENDA AT-A-GLANCE

Visual Studio / .NET Framework	Windows Client	Cloud Computing	Windows Phone	Cross-Platform Mobile Development	ASP.NET	JavaScript / HTML5 Client	SharePoint / Office	SQL Server

START TIME	END TIME	Visual Studio Live! Pre-Conference Workshops: Monday, October 6, 2014 (Separate entry fee required)			
7:30 AM	9:00 AM	Pre-Conference Workshop Registration			
9:00 AM	6:00 PM	**MW01** - Workshop: Deep Dive Into Visual Studio 2013, TFS, and Visual Studio Online - *Brian Randell*	**MW02** - Workshop: SQL Server for Developers - *Andrew Brust & Leonard Lobel*	**MW03** - Workshop: Data-Centric Single Page Applications with Angular, Breeze, and Web API - *Brian Noyes*	
6:00 PM	9:00 PM	Dine-A-Round Dinner			

START TIME	END TIME	Visual Studio Live! Day 1: Tuesday, October 7, 2014			
7:00 AM	8:00 AM	Registration			
8:00 AM	9:00 AM	Keynote: To Be Announced			
9:15 AM	10:30 AM	**T01** - Great User Experiences with CSS 3 - *Robert Boedigheimer*	**T02** - What's New in MVC 5 - *Miguel Castro*	**T03** - New IDE and Editor Features in Visual Studio 2013 - *Deborah Kurata*	**T04** - Creating Universal Windows Apps for Business - *Rockford Lhotka*
10:45 AM	12:00 PM	**T05** - Using jQuery to Replace the Ajax Control Toolkit - *Robert Boedigheimer*	**T06** - ASP.NET Reloaded: Web Forms vs. MVC vs. Web API - *Dino Esposito*	**T07** - Introduction to In Memory OLTP Using Hekaton in SQL Server 2014 - *Kevin Goff*	**T08** - WPF Data Binding in Depth - *Brian Noyes*
12:00 PM	1:30 PM	Lunch - Visit Exhibitors			
1:30 PM	2:45 PM	**T09** - Build an Angular and Bootstrap Web Application in Visual Studio from the Ground Up - *Deborah Kurata*	**T10** - What's New in Web API 2 - *Miguel Castro*	**T11** - Making the Most of the Visual Studio Online - *Brian Randell*	**T12** - Moving Web Apps to the Cloud - *Eric D. Boyd*
3:00 PM	4:15 PM	**T13** - JavaScript for the C# (and Java) Developer - *Philip Japikse*	**T14** - Never Mind the Mobile Web; Here's the Device Web - *Dino Esposito*	**T15** - Cross-platform Dev (iOS, Android and Java) with TFS and Team Explorer Everywhere - *Brian Randell*	**T16** - Solving Security and Compliance Challenges with Hybrid Clouds - *Eric D. Boyd*
4:15 PM	5:45 PM	Exhibitor Welcome Reception			

START TIME	END TIME	Visual Studio Live! Day 2: Wednesday, October 8, 2014			
7:00 AM	8:00 AM	Registration			
8:00 AM	9:00 AM	Keynote: To Be Announced			
9:00 AM	9:30 AM	Networking Break			
9:30 AM	10:45 AM	**W01** - Writing Next Generation JavaScript with TypeScript - *Rachel Appel*	**W02** - A Survey of Two Popular Visual Studio Tools - Web Essentials and NuGet - *John Petersen*	**W03** - Getting Started with Xamarin - *Walt Ritscher*	**W04** - SQL for Application Developers - Attendees Choose - *Kevin Goff*
11:00 AM	12:15 PM	**W05** - Building Single-Page Web Applications Using Kendo UI and the MVVM Pattern - *Ben Hoelting*	**W06** - ASP.NET MVC - Rudiments of Routing - *Walt Ritscher*	**W07** - Getting Started with Windows Phone Development - *Nick Landry*	**W08** - Database Development with SQL Server Data Tools - *Leonard Lobel*
12:15 PM	1:30 PM	Birds-of-a-Feather Lunch - Visit Exhibitors			
1:30 PM	2:45 PM	**W09** - Introduction to AngularJS - *John Petersen*	**W10** - Slice Development Time with ASP.NET MVC, Visual Studio, and Razor - *Philip Japikse*	**W11** - The Great Mobile Debate: Native vs. Hybrid App Development - *Nick Landry*	**W12** - Learning Entity Framework 6 - *Leonard Lobel*
3:00 PM	4:15 PM	**W13** - Creating Angular Applications Using Visual Studio LightSwitch - *Michael Washington*	**W14** - Build Data Driven Web Sites with WebMatrix 3 and ASP.NET Web Pages - *Rachel Appel*	**W15** - Busy .NET Developer's Guide to iOS - *Ted Neward*	**W16** - Team Foundation Server for Scrum Teams - *Richard Hundhausen*
4:30 PM	5:45 PM	**W17** - To Be Announced	**W18** - Build Real Time Websites and Apps with SignalR - *Rachel Appel*	**W19** - Busy .NET Developer's Guide to Android - *Ted Neward*	**W20** - Team Foundation Server: Must-Have Tools and Widgets - *Richard Hundhausen*

START TIME	END TIME	Visual Studio Live! Day 3: Thursday, October 9, 2014			
7:30 AM	8:00 AM	Registration			
8:00 AM	9:15 AM	**TH01** - Excel, Power BI and You: An Analytics Superhub - *Andrew Brust*	**TH02** - What's New in WPF 4.5 - *Walt Ritscher*	**TH03** - Visual Studio 2013, Xamarin and Windows Azure Mobile Services: A Match Made in Heaven - *Rick Garibay*	**TH04** - What's New in the .NET 4.5.1 BCL - *Jason Bock*
9:30 AM	10:45 AM	**TH05** - Big Data 101 with HDInsight - *Andrew Brust*	**TH06** - WPF for the Real World - *Brian Lagunas*	**TH07** - Building Your First Windows Phone 8.1 Application - *Brian Peek*	**TH08** - Essential C# 6.0 - *Mark Michaelis*
11:00 AM	12:15 PM	**TH09** - Cloud or Not, 10 Reasons Why You Must Know "Web Sites" - *Vishwas Lele*	**TH10** - Deploying WinRT Apps Without the Store - *Rockford Lhotka*	**TH11** - Building Games for Windows and Windows Phone - *Brian Peek*	**TH12** - To Be Announced
12:15 PM	1:15 PM	Lunch			
1:15 PM	2:30 PM	**TH13** - Loosely Coupled Applications with Service Bus and Document-centric Data Stores - *Vishwas Lele*	**TH14** - Creating Cross Platform Games with Unity - *Brian Lagunas*	**TH15** - Visual Studio Cloud Business Apps - *Michael Washington*	**TH16** - Asynchronous Debugging in .NET - *Jason Bock*
2:45 PM	4:00 PM	**TH17** - Not Your Father's BizTalk Server: Building Modern Hybrid Apps with Windows Azure BizTalk Services - *Rick Garibay*	**TH18** - Use Your .NET Code in WinRT with Brokered Assemblies - *Rockford Lhotka*	**TH19** - Best Practices: Building Apps for Office Using HTML/JavaScript - *Mark Michaelis*	**TH20** - Adventures in Unit Testing: TDD vs. TED - *Ben Hoelting*
4:15 PM	5:15 PM	Conference Wrap-Up - *Andrew Brust (Moderator), Jason Bock, Ben Hoelting, Rockford Lhotka, & Brian Peek*			

Sessions and speakers subject to change

TED NEWARD

Fun with C#, Part 2

Welcome back. In my last column, "Fun with C#" (msdn.microsoft.com/magazine/dn754595), I talked briefly about how familiarity with other programming languages can help clarify your thinking around a design problem that might otherwise seem intractable. I introduced a problem I encountered years ago during a consulting engagement in which I was required to reconcile a list of transactions stored locally with a supposedly equal list of transactions stored remotely. I had to either match them by transaction amount—nothing else was guaranteed to match for even the exact same transaction—or flag the unmatched items in the resulting list.

I chose to use F# for that exercise because it's a language with which I'm familiar. Quite frankly, it could easily have been another language like Scala, Clojure or Haskell. Any functional language would have worked in a similar fashion. The key here isn't the language itself or the platform on which it ran, but the concepts involved in a functional language. This was a pretty functional-friendly problem.

The F# Solution

Just to revisit, look at the F# solution in **Figure 1** before looking at how it would translate into C#.

Look at the previous column to see a brief recap of the F# syntax used in **Figure 1**, especially if you aren't familiar with F#. I'm also going to be going over it as I transform it into C#, so you could also just dive in.

Figure 1 **The F# Solution for Resolving Disparate Transactions**

```
type Transaction =
  {
    amount : float32;
    date : DateTime;
    comment : string
  }

type Register =
  | RegEntry of Transaction * Transaction
  | MissingRemote of Transaction
  | MissingLocal of Transaction

let reconcile (local : Transaction list) (remote : Transaction list) :
Register list =
  let rec reconcileInternal outputSoFar local remote =
    match (local, remote) with
    | [], _
    | _, [] -> outputSoFar
    | loc :: locTail, rem :: remTail ->
      match (loc.amount, rem.amount) with
      | (locAmt, remAmt) when locAmt = remAmt ->
        reconcileInternal (RegEntry(loc, rem) :: outputSoFar) locTail remTail
      | (locAmt, remAmt) when locAmt < remAmt ->
        reconcileInternal (MissingRemote(loc) :: outputSoFar) locTail remote
      | (locAmt, remAmt) when locAmt > remAmt ->
        reconcileInternal (MissingLocal(rem) :: outputSoFar) local remTail
      | _ ->
        failwith "How is this possible?"

  reconcileInternal [] local remote
```

The C# Solution

At the starting point, you need the Transaction and Register types. The Transaction type is easy. It's a simple structural type with three named elements, making it easy to model as a C# class:

```
class Transaction
{
  public float Amount { get; set; }
  public DateTime Date { get; set; }
  public String Comment { get; set; }
}
```

Those automatic properties make this class almost as short as its F# cousin. In all honesty, if I was going to really make it a one-to-one translation of what the F# version does, I should introduce overridden Equals and GetHashCode methods. For the purpose of this column, though, this will work.

Things get tricky with the discriminated union Register type. Like an enumeration in C#, an instance of a Register type can only be one of three possible values (RegEntry, MissingLocal or MissingRemote). Unlike a C# enumeration, each of those values in turn can contain data (the two Transactions that matched for RegEntry, or the missing Transaction for either MissingLocal or MissingRemote). While it would be easy to create three distinct classes in C#, these three classes must be somehow related. We need a List that can contain any of the three—but only those three—for the returned output, as shown in **Figure 2**. Hello, inheritance.

It's not ridiculously complicated, just more verbose. And if this were production-facing code, there are a few more methods I should add—Equals, GetHashCode and, almost certainly, ToString. Although there might be a few ways to make it more idiomatically C#, I'll write the Reconcile method fairly close to its F# inspiration. I'll look for idiomatic optimizations later.

The F# version has an "outer," publicly accessible function recursively called into an inner, encapsulated function. However, C# has no concept of nested methods. The closest I can approximate is with two methods—one declared public and one private. Even then,

Figure 2 **Use Inheritance to Include Three Distinct Classes**

```
class Register { }

  class RegEntry : Register
  {
    public Transaction Local { get; set; }
    public Transaction Remote { get; set; }
  }
  class MissingLocal : Register
  {
    public Transaction Transaction { get; set; }
  }
  class MissingRemote : Register
  {
    public Transaction Transaction { get; set; }
  }
```

this isn't quite the same. In the F# version, the nested function is encapsulated from everyone, even other functions in the same module. But it's the best we can get, as you can see in **Figure 3**.

As a side note, I can now accomplish the "hidden from everybody" recursive approach by writing the internal function entirely as a lambda expression referenced as a local variable inside Reconcile. That said, that's probably a little too much slavish adherence to the original and entirely not idiomatic to C#.

It isn't something most C# developers would do, but it would have almost the same effect as the F# version. Inside of ReconcileInternal, I have to explicitly extract data elements used. Then I explicitly write it in an if/else-if tree, as opposed to the more succinct and terse F# pattern match. However, it's really the exact same code. If either the local or remote list is empty, I'm done recursing. Just return the output and call it a day, like so:

```
static List<Register> ReconcileInternal(List<Register> Output,
                                        List<Transaction> local,
                                        List<Transaction> remote)
{
  if (local.Count == 0)
    return Output;
  if (remote.Count == 0)
    return Output;
```

Then, I need to extract the "heads" of each list. I also need to keep a reference to the remainder of each list (the "tail"):

```
Transaction loc = local.First();
List<Transaction> locTail = local.GetRange(1, local.Count - 1);

Transaction rem = remote.First();
List<Transaction> remTail = remote.GetRange(1, remote.Count - 1);
```

This is one place where I can introduce a huge performance hit if I'm not careful. Lists are immutable in F#, so taking the tail of a list is simply taking a reference to the second item in the list. No copies are made.

C#, however, has no such guarantees. That means I could end up making complete copies of the list each time. The GetRange method says it makes "shallow copies," meaning it will create a new List. However, it will point to the original Transaction elements. This is probably the best I can hope for without getting too exotic. Having said that, if the code becomes a bottleneck, get as exotic as necessary.

Looking again at the F# version, what I'm really examining in the second pattern match is the amounts in the local and remote transaction, as shown in **Figure 4**. So I extract those values as well, and start comparing them.

Each branch of the tree is pretty easy to understand at this point. I add the new element to the Output list, then recurse with the unprocessed elements of the local and remote lists.

Figure 3 **The Nested Function Is Encapsulated Here**

```
class Program
{
  static List<Register> ReconcileInternal(List<Register> Output,
                                          List<Transaction> local,
                                          List<Transaction> remote)
  {
    // . . .
  }
  static List<Register> Reconcile(List<Transaction> local,
                                  List<Transaction> remote)
  {
    return ReconcileInternal(new List<Register>(), local, remote);
  }
}
```

Figure 4 **The F# Version Examines the Local and Remote Amounts**

```
  float locAmt = loc.Amount;
  float remAmt = rem.Amount;

  if (locAmt == remAmt)
  {
    Output.Add(new RegEntry() { Local = loc, Remote = rem });
    return ReconcileInternal(Output, locTail, remTail);
  }
  else if (locAmt < remAmt)
  {
    Output.Add(new MissingRemote() { Transaction = loc });
    return ReconcileInternal(Output, locTail, remote);
  }
  else if (locAmt > remAmt)
  {
    Output.Add(new MissingLocal() { Transaction = rem });
    return ReconcileInternal(Output, local, remTail);
  }
  else
    throw new Exception("How is this possible?");
}
```

Wrapping Up

If the C# solution is really this elegant, why bother taking the stop through F# in the first place? It's hard to explain unless you go through the same process. Essentially, I needed the stop through F# to flesh out the algorithm in the first place. My first attempt at this was an absolute disaster. I started by iterating through the two lists using double "foreach" loops. I was trying to track the state along the way, and I ended up with a huge, steaming mess I would never have been able to debug in a million years.

Learning how to "think differently" (to borrow a famous computer company's marketing line from a few decades ago) yields results, not the choice of language itself. I could've easily told this story going through Scala, Haskell or Clojure. The point wasn't the language feature set, but the concepts behind most functional languages—recursion, in particular. That's what helped break through the mental logjam.

This is part of the reason developers should learn a new programming language every year, as first suggested by one of the Pragmatic Programmers, Dave Thomas, of Ruby fame. Your mind can't help but be exposed to new ideas and new options. Similar kinds of ideas emerge when a programmer spends some time with Scheme, Lisp or with a stack-based language such as Forth—or with a prototype-based language like Io.

If you'd like a lightweight introduction to a number of different languages all off the Microsoft .NET Framework platform, I highly recommend Bruce Tate's book, "Seven Languages in Seven Weeks" (Pragmatic Bookshelf, 2010). Some of them you wouldn't use directly on the .NET platform. Then again, sometimes the win is in how we think about the problem and frame the solution, not necessarily reusable code. Happy coding! ■

Ted Neward *is the CTO at iTrellis, a consulting services company. He has written more than 100 articles and authored and coauthored a dozen books, including "Professional F# 2.0" (Wrox, 2010). He's a C# MVP and speaks at conferences around the world. He consults and mentors regularly—reach him at ted@tedneward.com or ted@itrellis.com if you're interested in having him come work with your team, and read his blog at blogs.tedneward.com.*

Thanks to the following Microsoft technical expert for reviewing this article: *Lincoln Atkinson*

RACHEL APPEL

Build Universal Apps for the Windows Platform

If you develop for the Windows platform, you can now write apps that target both Windows 8.1 and Windows Phone 8.1 with a single, shared codebase by creating a Universal App. Universal Apps are new to Windows. They let you share significant portions of your JavaScript, C#/Visual Basic, and C++ code across the Windows ecosystem.

At the Build 2014 conference, Microsoft stated it will extend the Universal App concept to Xbox in the future (bit.ly/1p19070). You can do it now, but you'd need Visual Studio 2012 and a few workarounds. This is particularly exciting news for those who build games and media apps. You now have a head start building for the various Windows platforms, and adding an Xbox project later. Here, I'll stick to Windows and Windows Phone.

Outside the Windows and Xbox platforms, you can use Universal Apps to write cross-platform apps with HTML and JavaScript. You can write strictly HTML5. You can eschew the Windows Library for JavaScript (WinJS) for iOS and Android. You can also use WinJS just for native features in the Windows ecosystem. If you're using XAML with C#, you can use a tool like Xamarin to publish across the major platforms.

Get Started with Universal Apps

You can build Universal Apps using Visual Studio 2013 Update 2 on Windows 8.1 using project templates for JavaScript, C#/Visual Basic and C++. This means you can continue to use your favorite language to build apps in this latest Windows update.

The structure of the new Universal App is a Visual Studio solution with at least three projects—Windows, Windows Phone and Shared. Coincidentally, there are three project types of Universal Apps from which to choose—the Blank, Hub and Navigation templates. You can use the New Project dialog in Visual Studio to create a Universal App with any of these three new templates.

Because Visual Studio creates a solution containing three projects, you would rightfully expect the Shared project is where the shared code goes. Put as much code here as possible. Code sharing and reuse has many benefits, such as easier maintenance and less-expensive bug fixes. Of course, you can have too much of a good thing. If you try to share everything, you'll end up with too many branching constructs in your code, such as if and switch statements that will become unruly. As a rule of thumb, if you find yourself copying and pasting code to use in multiple places, you should separate the code into the separate projects. This concept is called DRY—short for Don't Repeat Yourself. The DRYer the code is, the better.

A Shared project is a project file with a .shproj extension. Shared projects behave like regular projects, except they don't create compiled output or a package. They're simply a way to make code available for two or more projects without having to reference or copy and paste. You can share code like this because in Windows 8.1, the Windows Runtime extends across both OSes. Some API members aren't compatible between platforms, but calls to those can go in their corresponding projects.

Figure 1 shows the architecture for a simple Universal App. This app, called Countdown, displays the number of days to an event date the user enters alongside number of days until the event.

As you can see in **Figure 1**, each project contains its own UI code. The Shared project contains a few JavaScript files, as well as a shared home page.

Sharing Code Across Projects

Each platform project has a default.html and .js file that acts as the app starting point. These reference the corresponding platform-specific JavaScript files. For this example, you don't need any special code for lifecycle management. If you do, though, you might need to separate those files depending on how OS-specific your needs are for the app.

These are the Windows default.html references:

```
<script src="//Microsoft.WinJS.2.0/js/base.js"></script>
<script src="//Microsoft.WinJS.2.0/js/ui.js"></script>
```

These are the Windows Phone default.html references:

```
<script src="//Microsoft.Phone.WinJS.2.1/js/base.js"></script>
<script src="//Microsoft.Phone.WinJS.2.1/js/ui.js"></script>
```

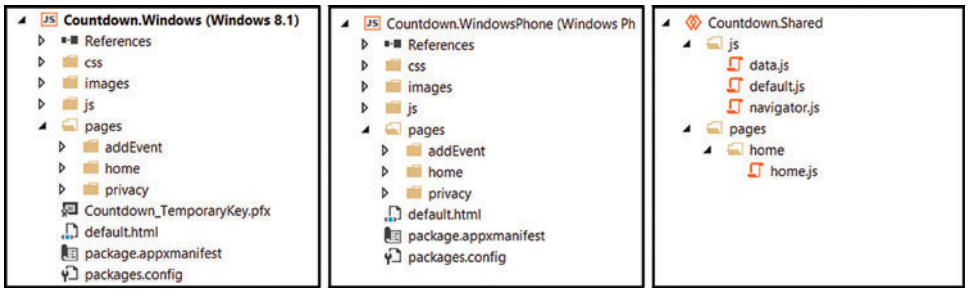


Figure 1 **Side-by-Side Views of Windows 8.1, Windows Phone 8.1 and a Shared Project**

Figure 2 **The HTML and CSS that Create the Home Page for Both Projects**

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Countdown</title>
  <link href="/pages/home/home.css" rel="stylesheet" />
  <script src="/js/MobileServices.js"></script>
  <script src="/js/data.js"></script>
  <script src="/pages/home/home.js"></script>
</head>
<body>
  <div id="contenthost">
    <h1 id="title" class="title">Countdown from</h1>
    <div id="maincontent">
      <div id="listViewTemplate" data-win-control="WinJS.Binding.Template"
        class="win-container">
        <div data-win-bind="style.background: color" class="win-item">
          <h1 class="h1" data-win-bind="textContent: daysToGo"></h1>
          <h2 class="h2"> days to go until</h2>
          <h2 class="h2" data-win-bind="textContent: eventTitle"></h2>
          <h2 class="h2" data-win-bind="textContent: eventDate"></h2>
        </div>
      </div>
      <div id="listView" data-win-control="WinJS.UI.ListView" class="win-listview"
           data-win-options="{ selectionMode: 'single' }">
      </div>
    </div>
    <div id="addEventHTMLControl">
    </div>
  </div>
</body>
</html>

<!-- CSS for the Windows  ListView -->
#listView {
height:500px;
}

#listView .win-viewport {
  height:415px;
}

#listView .win-item {
  width:300px;
  height:185px;
  padding: 0 0 0 5px;
}

.h1 {
  font-size:1.5em;
}

.h2 {
  font-size:1em;
}

<!-- CSS for the Windows.Phone ListView -->
#listView .win-item {
  width:100%;
  height:125px;
  padding: 5px 0 0 5px;
}
.h1 {
  font-size:2em;
}

.h2 {
  font-size:1.5em;
}
```

These are references to the WinJS 2.0 for Windows and the WinJS 2.1 for Windows Phone. The latter contains the JavaScript that takes advantage of any differences on Windows Phone.

In the sample Countdown app, there's a page directory in each platform project. Each directory contains subdirectories for each page. The subdirectories include the page itself and related items such as CSS or JS files. In **Figure 1**, you can see the pages are home, addEvent and privacy. These folders merge between the platform and shared projects at run time. You may only have one file with a unique name in each folder. For example, you can save the home.html page in the /pages/home/ directory, but only in one project. In this case, I've put it in the Shared project folder.

The home page is the main part of the app where the countdown events display. **Figure 2** shows the shared UI code for home.html. Notice the corresponding home.css files are located in their respective projects. This means if you use the same CSS class names across projects, you can use the same HTML. But these would be restyled with the varying CSS for each platform, as you can see in **Figure 2**. You can have one base of HTML and style with differing CSS for each platform project.

Figure 3 converts the element from **Figure 2** into a WinJS.UI.HtmlControl on Windows. This means you can store the add event code in the addEvent.html file.

Within the shared home.js, there are a few points where you need to write some conditional code. This determines which platform you're using and acts accordingly. Check for the package name to determine which platform is running. The package name is in the package.appmanifest under the Packaging tab. By default, this is a GUID. You can change it to something friendly to use in code. Once you've done that, you can query it to determine the current platform at run time. **Figure 3** shows a sample of what this code looks like when determining which events to wire up and which UI component to use.

As you can see, **Figure 3** wires the appBar buttons to features that are distinct to each app type. This could be a Flyout versus an entire page for both adding events and viewing privacy settings. It's easy in Windows. The privacy page is part of the app settings, launched from the Settings charm. The code that sets up Charms usually goes in default.js on activated event. Because Windows Phone doesn't have the notion of Charms, use the appBar as a way to navigate to the privacy page, as reflected in **Figure 3**.

The best way to add events is to use a Flyout in Windows. Then, due to the differing form factors, create a complete HTML page on Windows Phone. Popups and dialogs just don't do well on phone-sized

Figure 3 **Part of the Home.js Ready Function That Determines the OS and Builds the UI**

```
var listView = document.querySelector("#listView").winControl;
var packageName = Windows.ApplicationModel.Package.current.id.name;
if (packageName === "Countdown.WindowsPhone") {
  document.querySelector("#privacyButton").addEventListener("click",
    this.navigateToPrivacyPage);
  document.querySelector("#addButton").addEventListener("click",
    this.navigateToaddEventPage);
  listView.layout = new WinJS.UI.ListLayout();
}
else if (packageName === "Countdown.Windows") {
  document.querySelector("#addButton").addEventListener("click",
    this.showAddEventFlyout);
  var htmlControl =
    new WinJS.UI.HtmlControl(document.querySelector("#addEventHTMLControl"),
    { uri: '/pages/addEvent/addEvent.html' });
  listView.layout = new WinJS.UI.GridLayout();
}
```

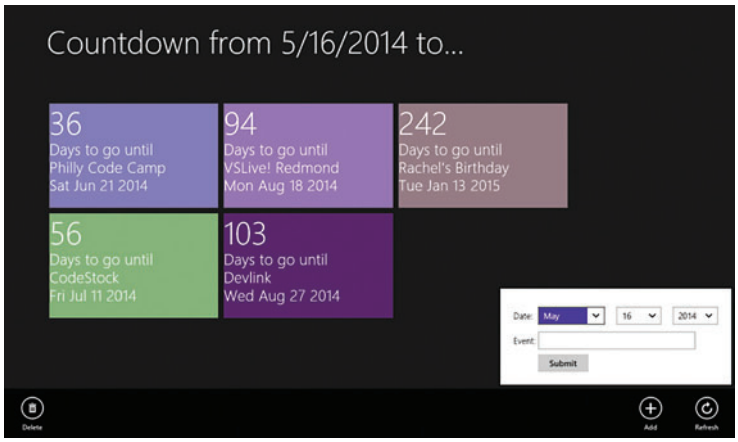


Figure 4 **The Runtime Appearance of the Countdown App Flyout**

devices. The addEvent Flyout or page is where the user enters new event details. The HTML for the addEvent Flyout is also in **Figure 3**.

Also notice in **Figure 3** the ListView layout changes between a grid and list layout in the code, depending on which platform is detected. The ListLayout property is handy, because you need just one block of HTML from **Figure 2**. Combining this with one shared chunk of code from **Figure 3** does the legwork to switch between layouts.

In the js folder, you'll find the same data.js and navigator.js files as their counterparts in any given solution based on the Navigation template. Apps on both platforms need the same data and navigation scheme, so these files can stay in the Shared project. The following code is the definition for the Data namespace, which is familiar to WinJS developers:

```
var list = new WinJS.Binding.List();

WinJS.Namespace.define("Data", {
  loadEventData: loadEventData,
  saveEventItem: saveEventItem,
  deleteEventItem: deleteEventItem,
  items: list,
});
```

Both platform projects consume the data the exact same way, by JSON data put into a WinJS.Binding.List object. The only difference is how they style the data.

When developing Universal Apps, you should write unit tests and make changes in the shared codebase. This means at some point, you'll need to run the project to verify the results. When debugging, choose the project you want to debug, right-click on that project in the Solution Explorer and choose Set as Start Project.

This will make the project appear bold in Solution Explorer. When you click Run, press F5, or launch the project, then the Start Project is the one that runs. You can switch between projects as the starting project as needed.

The Shared project can't be the Start Project. When you run a specific project, Visual Studio packages, deploys and loads that project's assets, as well as the ones from the Shared project. Then it starts the corresponding emulator if it isn't already and you're good to go. To learn more about debugging Windows Store apps, see my blog post at bit.ly/1hTpxHB.

Create Native UIs

The UI will share the least amount of code. Various form factors dictate both the amount and presentation of data. Present data with HTML and make it look good with CSS. Dealing with different HTML and CSS is simpler if you split it into separate projects, including the CSS Media Queries for each range of devices per platform.

In the case of the Countdown app, you want a grid display on Windows and a vertical list on phones. Do this by placing the home.html file in the Shared project, and the home.css files in separate projects. The addEvent and privacy pages go in their corresponding platform projects. To see this folder structure, look back at **Figure 1**.

In the Countdown.Windows app, the user adds an event by tapping or clicking the Add appBar button. This displays a popup dialog using a Flyout control. In Countdown.WindowsPhone, the app takes the user to an entire addEvent page. Here's the code that designs the Flyout in Countdown.Windows:

```
<div id="eventFlyoutPanel" data-win-control="WinJS.UI.Flyout">
  <table width="100%" height="100%">
    <tr><td>Date:</td><td><span id="eventDate"
      data-win-control="WinJS.UI.DatePicker"></span></td></tr>
    <tr><td>Event:</td><td><input type="text" id="eventTitle" /></td></tr>
    <tr><td> </td><td><input type="button" id="confirmButton"
      value="Submit" /></td></tr>
  </table>
</div>
```

It's a small table containing a DatePicker and TextBox. Neither Flyouts nor DatePickers run on Windows Phone. This Flyout is configured in home.html as a WinJS.UI.HtmlControl. **Figure 4** gives you a look at the runtime appearance of the screen this code creates.

The Countdown.WindowsPhone project is much different. Instead of a Flyout, you must navigate to the addEvent page. Having no DatePicker is no problem. You can collect the event's date with the proven mobile technique of creating three <select> elements, one for each part of the date (Day/Month/Year). **Figure 5** contains a sample of the Countdown.Windows addEvent HTML code and **Figure 6** shows what it will look like at run time on Windows Phone.

You need to ensure the controls you want to use are available across OSes. Hub, Flyout, DatePicker and TimePicker controls aren't available on Windows Phone. The Pivot control isn't available on Windows. Fortunately, if you're using the popular Hub or Pivot controls, they're usually interchangeable. So you can use Hub on Windows and Pivot on the phone and get the same things done. Simple dropdown controls work for capturing dates. **Figure 6** shows what this will look like on the Windows phone.

Review your code and separate code targeting OS-specific tasks. The more of these types of tasks, the more it should be separated into its own project. If there are only a few blocks, then it's OK to leave it in the Shared project.

More Ways to Share Code

You can apply architectural patterns as a way to organize, maintain and share code, such as Model-View-ViewModel (MVVM) or Model-View-Controller (MVC). In the case of the Countdown app, sharing code in the Shared project works fine.

Another exciting announcement from the Build conference is that the Windows Runtime now supports any third-party JavaScript framework. This means you can use popular packages such as Knockout, Angular or Breeze.js to implement these patterns.

The Models and ViewModels in Shared project work seamlessly across the other projects with little or no modification. This leaves only code in the Views that may need duplication or customization. Sometimes this is unavoidable. There are vastly different device sizes and form factors to consider. There are also wide varieties of input devices including touch, keyboard, mouse, voice and so on. Code to handle these variables often needs to be separate.

In JavaScript apps, you can put Process Lifecycle Management code in the default.js in each individual project. You can also store models, ViewModels and any app logic there. The OSes handle protocol activations differently. However, most projects have some general-purpose utility code, which should be added to a script in the Shared project.

There are several other ways to share code across projects:

- **Conditional Compilation:** Segment blocks of code to compile specifically for one OS or another. This applies to C#/Visual Basic and C++. Use regular if statements in WinJS. Learn more about this at bit.ly/UzdBAa.
- **Runtime Components:** Components built with the Windows Runtime expose code to both Windows and Windows Phone apps, which allows code sharing.
- **Add Link:** You can treat a pointer to a file in another project as if it were part of the current project.

Figure 5 **A Sample of the Countdown.Windows addEvent HTML Code**

```
<h1 id="title" class="title">
  Add event
  </h1>
  <div>
  Month
    <select id="eventMonth">
    <option value="--Select--">--Select--</option>
    <option value="01">January</option>
    <option value="02">February</option>
    <option value="03">March</option>
    <option value="04">April</option>
    <option value="05">May</option>
    <!-- options 6-11, cut for brevity -->
    <option value="12">December</option>
  </select>
  Day
  <select name="day" id="day">
    <option value="--Select--">--Select--</option>
    <option value="01">01</option>
    <option value="02">02</option>
    <option value="03">03</option>
    <option value="04">04</option>
    <option value="05">05</option>
    <!-- options 6-30, cut for brevity -->
    <option value="31">31</option>
  </select>
  Year<select id="eventYear">
    <option value="--Select--">--Select--</option>
    <!-- all other options created via client script -->
    </select>
  </div>
  <div>Event<input type="text" id="eventTitle" /></div>
  <div>
    <input type="button" id="backButton" value="Back" />
    <input type="button" id="confirmButton" value="Submit" />
  </div>
<script type="text/javascript">
  (function () {
    var yearSelect = document.querySelector("#eventYear");
    for (var i = 2014; i < 2075; i++) {
      yearSelect.options[i] = new Option(i, i);
    }
  })();
</script>
```

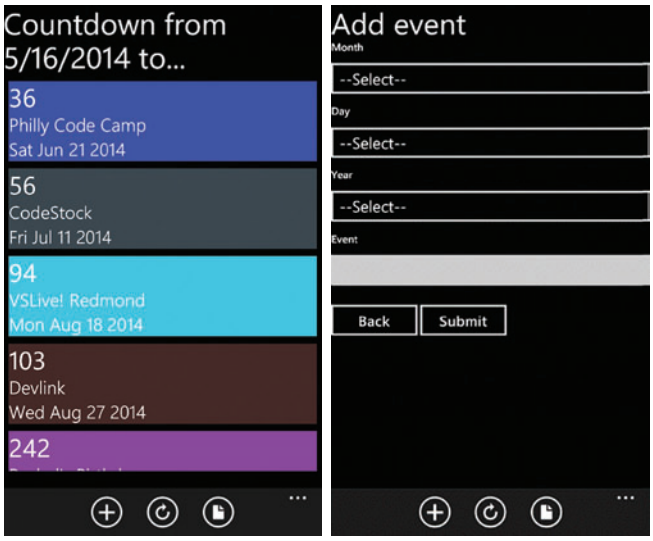


Figure 6 **The Countdown and Add Event Page as Generated on a Windows Phone**

- **Portable Class Library (PCL):** This is the Microsoft .NET Framework way to share code. The Windows Runtime (WinRT) supports these legacy components, so you can use them in your apps. These are best to use if you already have an existing PCL. You might want to add WinRT components if you're creating new components.

Clearly, a Shared project isn't the only way to share code. You can reuse .NET code you already have lying around. For example, you may have some hefty performance requirements when creating a live action game that needs the help of C++. UsingWinRT components is a great way to share that kind of code.

Use Fragments whenever possible. These are just snippets of HTML both projects can use. UI components such as appBars and Flyouts don't work well across form factors, so you'll want to keep those in their corresponding projects. Code that sets general app settings and the like can go in the Shared project.

Wrapping Up

While Universal Apps don't exactly provide a "write once and magically run everywhere" scenario, they do come close. There are some API changes, but with the single Windows Runtime behind the apps, it's quite easy to work on an app that runs in the entire Windows ecosystem. You can even get your apps running on non-Microsoft platforms with the help of Visual Studio extensions from Xamarin. Also promised at Build was future support for Xbox One Achievements, Challenges, OneGuide and more. There are exciting things happening for Microsoft app developers. ■

Rachel Appel *is a consultant, author, mentor and former Microsoft employee with more than 20 years of experience in the IT industry. She speaks at top industry conferences such as Visual Studio Live!, DevConnections, MIX and more. Her expertise lies within developing solutions that align business and technology focusing on the Microsoft dev stack and open Web. For more about Appel, visit her Web site at rachelappel.com.*

Thanks to the following Microsoft technical expert for reviewing this article: *Frank La Vigne*

CHARLES PETZOLD

Vertex Shaders and Transforms

In 1975, the artist Richard Haas painted the flat side of a building in the Soho district of Manhattan to resemble the façade of a classic cast-iron building, including such features as a cat in a window. Time hasn't been kind to this painting, but in its early days, it looked very real and fooled many people.

Such paintings—called trompe-l'œil, meaning "deceive the eye"—use shadows and shading to bring out a third dimension on a flat two-dimensional surface. We tend to be tickled by such illusions, but at the same time eager to satisfy ourselves that we can discern the trick. One simple way is to try observing the painting from different perspectives to see if it looks the same.

A similar mental process occurs when a computer program displays some graphics that seem to straddle the line between 2D and 3D. Is it really 3D? Or just 2D with some clever overlapping and shading? We can't move our heads from side to side to establish the facts, but we might be able to persuade the program to rotate its graphics to see what happens.

The image in **Figure 1** looks a lot like the screen displayed by the ThreeTriangles program in the previous installment of this column (msdn.microsoft.com/magazine/dn768854). But the downloadable program for *this* column, called ThreeRotatingTriangles, does indeed rotate that assemblage of three triangles. The effect is visually quite interesting, and as the three triangles move in relation to each other, the program does establish that there is indeed some 3D graphics processing going on. However, if you look at the code, you'll discover that the program is still written entirely with Direct2D rather than Direct3D, using the powerful feature of Direct2D effects.

Getting Data into an Effect

The overall organization of the earlier ThreeTriangles program has been preserved in ThreeRotatingTriangles. The class that provides the Direct2D effect implementation is now named RotatingTriangleEffect rather than SimpleTriangleEffect, but it continues to implement the ID2D1EffectImpl ("effect implementation") and ID2D1DrawTransform interfaces.

The earlier SimpleTriangleEffect wasn't versatile at all. It contained hardcoded vertices to display three overlapping triangles. RotatingTriangleEffect allows the vertices to be defined from outside the class, and both the effect implementation and the vertex shader have been enhanced to accommodate matrix transforms.

Code download available at msdn.microsoft.com/magazine/msdnmag0914.

Generally, an effect implementation such as RotatingTriangleEffect contains a static method that registers itself by calling RegisterEffectFromString and associating itself with a class ID. In the ThreeRotatingTriangles program, the ThreeRotatingTrianglesRenderer class calls this static method in its constructor to register the effect.

ThreeRotatingTrianglesRenderer also defines an object of type ID2D1Effect as a private field:

```
Microsoft::WRL::ComPtr<ID2D1Effect>
          m_rotatingTriangleEffect;
```

To use the effect, the program must create this object by referencing the effect's class ID in a call to CreateEffect. In the ThreeRotatingTrianglesRenderer class, this occurs in the CreateDeviceDependentResources method:

```
d2dContext->CreateEffect(
  CLSID_RotatingTriangleEffect, &m_rotatingTriangleEffect);
```

The effect can then be rendered with a call to the DrawImage method. Here's how ThreeRotatingTrianglesRenderer makes the call in its Render method:

```
d2dContext->DrawImage(m_rotatingTriangleEffect.Get());
```

But there's more that can be done between those two calls. ID2D1Effect derives from ID2D1Properties, which has methods named SetValue and GetValue that allow a program to set properties on the effect. These properties can range from simple effect options expressed as Boolean flags to large buffers of data. However, SetValue and GetValue aren't often used. They require identifying the particular property by index, and greater program clarity is obtained by instead using the methods SetValueByName and GetValueByName.

Keep in mind that these SetValueByName and GetValueByName methods are part of the ID2D1Effect object, which is the object returned from the CreateEffect call. The ID2D1Effect object passes

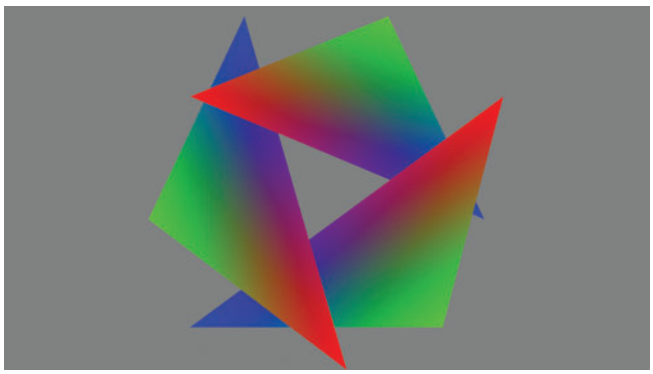


Figure 1 **The ThreeRotatingTriangles Program Display**

Figure 2 **Creating the Effect and Setting the Vertex Buffer**

```
void ThreeRotatingTrianglesRenderer::CreateDeviceDependentResources()
{
  ID2D1DeviceContext1* d2dContext =
    m_deviceResources->GetD2DDeviceContext();

  // Create the effect
  DX::ThrowIfFailed(d2dContext->CreateEffect(
                  CLSID_RotatingTriangleEffect,
                  &m_rotatingTriangleEffect)
    );

  // Set the vertices
  std::vector<PositionColorVertex> vertices =
  {
    // Triangle 1
    { XMFLOAT3(0, -1000, -1000), XMFLOAT3(1, 0, 0) },
    { XMFLOAT3(985, -174, 0), XMFLOAT3(0, 1, 0) },
    { XMFLOAT3(342, 940, 1000), XMFLOAT3(0, 0, 1) },

    // Triangle 2
    { XMFLOAT3(866, 500, -1000), XMFLOAT3(1, 0, 0) },
    { XMFLOAT3(-342, 940, 0), XMFLOAT3(0, 1, 0) },
    { XMFLOAT3(-985, -174, 1000), XMFLOAT3(0, 0, 1) },

    // Triangle 3
    { XMFLOAT3(-866, 500, -1000), XMFLOAT3(1, 0, 0) },
    { XMFLOAT3(-643, -766, 0), XMFLOAT3(0, 1, 0) },
    { XMFLOAT3(643, -766, 1000), XMFLOAT3(0, 0, 1) }
  };

  DX::ThrowIfFailed(
    m_rotatingTriangleEffect->SetValueByName(L"VertexData",
      (byte *) &vertices)
    );

  // Ready to render!
  m_readyToRender = true;
}
```

the values of these properties to the effect implementation class you've written—the class that implements the ID2D1EffectImpl and ID2D1DrawTransform interfaces. This seems a bit roundabout, but it's done this way so registered effects can be used without access to the classes that implement the effect.

But this means that an effect implementation such as the RotatingTriangleEffect class must itself indicate that it can accept properties of various types, and it must provide methods for setting and getting those properties.

This information is provided by the effect implementation when it registers itself using RegisterEffectFromString. Required in this call is some XML that includes the names and types of the various properties the effect implementation supports. RotatingTriangleEffect supports four properties, with the following names and data types:

- VertexData of type blob, that is, a memory buffer referenced by a byte pointer.
- ModelMatrix of type matrix4x4.
- ViewMatrix of type matrix4x4.
- ProjectionMatrix of type matrix4x4.

The names of these data types are specific to Direct2D effects. When an effect that supports properties is registered, the effect implementation must also supply an array of D2D1_VALUE_TYPE_BINDING objects. Each of the objects in this array associates a named property, for example VertexData, with two methods in the effect implementation that set and get the data. For VertexData, the two methods are named SetVertexData and GetVertexData. (When you define these Get methods, make sure to include the const keyword, or you'll get one of those weird template errors that will be completely baffling.)

Similarly, the RotatingTriangleEffect class defines methods named SetModelMatrix and GetModelMatrix, and so forth. These methods aren't called by any application program—indeed, they're private to RotatingTriangleEffect. Instead, the program calls the SetValueByName and GetValueByName methods on the ID2D1Effect object, which then calls the Set and Get methods in the effect implementation.

The Vertex Buffer

The ThreeRotatingTrianglesRenderer class registers the RotatingTrianglesEffect in its constructor, and renders the effect in its Render method. But between those two calls, the renderer class calls SetValueByName on the ID2D1Effect object to pass data into the effect implementation.

The first of the four effect properties listed earlier is VertexData, which is a collection of vertices used to define a vertex buffer. For Direct2D effects, the number of items in a vertex buffer must be a multiple of three, grouped into triangles. The ThreeRotatingTriangles program displays only three triangles with three vertices each, but the effect can handle a larger buffer.

The format of the vertex buffer expected by RotatingTriangleEffect is defined in a structure in RotatingTriangleEffect.h:

```
struct PositionColorVertex
{
  DirectX::XMFLOAT3 position;
  DirectX::XMFLOAT3 color;
};
```

This is the same format used by SimpleTriangleEffect, but defined somewhat differently. **Figure 2** shows how the CreateDeviceDependentResources method in ThreeRotatingTrianglesRenderer transfers the vertex array to the effect after the effect has been created.

> For Direct2D effects, the number of items in a vertex buffer must be a multiple of three, grouped into triangles.

The X and Y coordinates are based on sines and cosines of angles in 40-degree increments with a radius of 1,000. The Z coordinates range from –1,000 for the foreground to 1,000 for the background. In the earlier SimpleTriangleEffect I was very careful to set Z between 0 and 1 because of the conventions used to clip 3D output. As you'll see, that's not necessary here because actual camera transforms will be applied to the vertices.

The SetValueByName call with a name of VertexData causes the ID2D1Effect object to call the SetVertexBuffer method in RotatingTriangleEffect to pass along the data. This method casts the byte pointer back to its original type and calls CreateVertexBuffer to store the information in a way that allows it to be passed to the vertex shader.

Applying the Transforms

Figure 3 shows the Update method in ThreeRotatingTriangles-Renderer calculating three matrix transforms and making three calls to SetValueByName. The code has been slightly simplified to remove checks for errant HRESULT returns.

As usual, Update is called at the frame rate of the video display. The first matrix it calculates is applied to the vertices to rotate them around the Y axis. The second matrix is a standard camera view transform that results in shifting the scene so the viewer is on the origin of the three-dimensional coordinate system and looking straight along the Z axis. The third is a standard projection matrix, which results in X and Y coordinates being normalized to values between –1 and 1, and Z coordinates between 0 and 1.

> As usual, Update is called at the frame rate of the video display.

These matrices must be applied to vertex coordinates. So why doesn't the program just multiply them by the array of vertices defined in the CreateDeviceDependentResources method and then set a new vertex buffer in the RotatingTrianglesEffect?

It's certainly possible to define a dynamic vertex buffer that changes at the frame rate of the video display, and in some cases it's necessary. But if the vertices need only be modified by matrix transforms, a dynamic vertex buffer isn't as efficient as maintaining the same buffer throughout the program and applying the transforms later on in the pipeline—specifically, in the vertex shader that's running on the video GPU.

Figure 3 **Setting the Three Transform Matrices**

```
void ThreeRotatingTrianglesRenderer::Update(DX::StepTimer const& timer)
{
  if (!m_readyToRender)
      return;

  // Apply model matrix to rotate vertices
  float angle = float(XM_PIDIV4 * timer.GetTotalSeconds());
  XMMATRIX matrix = XMMatrixRotationY(angle);
  XMFLOAT4X4 float4x4;
  XMStoreFloat4x4(&float4x4, XMMatrixTranspose(matrix));
  m_rotatingTriangleEffect->SetValueByName(L"ModelMatrix", float4x4);

  // Apply view matrix
  matrix = XMMatrixLookAtRH(XMVectorSet(0, 0, -2000, 0),
                            XMVectorSet(0, 0, 0, 0),
                            XMVectorSet(0, 1, 0, 0));
  XMStoreFloat4x4(&float4x4, XMMatrixTranspose(matrix));
  m_rotatingTriangleEffect->SetValueByName(L"ViewMatrix", float4x4);

  // Base view width and height on coordinates of model
  float width = 2000;
  float height = 2000;

  // Adjust width and height for landscape and portrait modes
  Windows::Foundation::Size logicalSize = m_deviceResources->GetLogicalSize();

  if (logicalSize.Width > logicalSize.Height)
      width *= logicalSize.Width / logicalSize.Height;
  else
      height *= logicalSize.Height / logicalSize.Width;

  // Apply projection matrix
  matrix = XMMatrixOrthographicRH(width, height, 500, 4000);
  XMStoreFloat4x4(&float4x4, XMMatrixTranspose(matrix));
  m_rotatingTriangleEffect->SetValueByName(L"ProjectionMatrix", float4x4);
}
```

This means the vertex shader needs new matrix transforms for every frame of the video display, and that raises another issue: How does an effect implementation get data into the vertex shader?

The Shader Constant Buffer

Data is transferred from application code into a shader through a mechanism called a constant buffer. Don't let the name deceive you into thinking its contents remain constant throughout the course of the program. That's definitely not the case. Very often the constant buffer changes with every frame of the video display. However, the contents of the constant buffer are constant for all the vertices in each frame, and the format of the constant buffer is fixed at compile time by the program.

The format of the vertex shader constant buffer is defined in two places: in C++ code and in the vertex shader itself. In the effect implementation, it looks like this:

```
struct VertexShaderConstantBuffer
{
  DirectX::XMFLOAT4X4 modelMatrix;
  DirectX::XMFLOAT4X4 viewMatrix;
  DirectX::XMFLOAT4X4 projectionMatrix;
} m_vertexShaderConstantBuffer;
```

When the Update method in the renderer calls SetValueByName to set one of the matrices, the ID2D1Effect object calls the appropriate Set method in the RotatingTrianglesEffect class. These methods are named SetModelMatrix, SetViewMatrix, and SetProjection-Matrix, and they simply transfer the matrix to the appropriate field in the m_vertexShaderConstantBuffer object.

The ID2D1Effect object assumes that any call to SetValueByName results in a change to the effect that probably involves a corresponding change to graphical output, so the PrepareForRender method is called in the effect implementation. It's here the effect implementation can take the opportunity to call SetVertexShaderConstantBuffer to transfer the contents of the VertexShaderConstantBuffer to the vertex shader.

The New Vertex Shader

Now, finally, you can look at the High Level Shader Language (HLSL) code that's doing much of the work in rotating the vertices of those three triangles and orienting them in 3D space. This is the new and improved vertex shader, shown in its entirety in **Figure 4**.

Notice how the structures are defined: The VertexShaderInput in the shader is the same format as the PositionColorVertex structure defined in the C++ header file. The VertexShaderConstantBuffer is the same format as the same-named structure in C++ code. The VertexShaderOutput structure matches the PixelShaderInput structure in the pixel shader.

In the vertex shader associated with the SimpleTriangleEffect in last month's column, a ClipSpaceTransforms buffer was provided automatically to convert from scene space (the pixel coordinates used for the vertices of the triangles) to clip space, which involves normalized X and Y coordinates ranging from –1 to 1, and Z coordinates that range from 0 to 1.

That's no longer necessary, so I removed it without any unfortunate consequences. Instead, the projection matrix does the equivalent job. As you can see, the main function applies the three matrices to the input vertex position, and sets the result to the clipSpaceOuput field of the output structure.

That clipSpaceOutput field is required. This is how the depth buffer is managed, and how the results are mapped to the display surface. However, the sceneSpaceOutput field of the VertexShaderOutput structure isn't required. If you remove that field—and also remove the field from the PixelShaderInput structure of the pixel shader—the program will run the same.

Row Major and Column Major

The shader performs three multiplications of positions by matrices:

```
pos = mul(pos, modelMatrix);
pos = mul(pos, viewMatrix);
pos = mul(pos, projectionMatrix);
```

In mathematical notation, these multiplications look like this:

$$\begin{vmatrix} x & y & z & w \end{vmatrix} \times \begin{vmatrix} m11 & m12 & m13 & m14 \\ m21 & m22 & m23 & m24 \\ m31 & m32 & m33 & m34 \\ m41 & m42 & m43 & m44 \end{vmatrix}$$

When this multiplication is performed, the four numbers that describe the point (*x, y, z, w*) are multiplied by the four numbers in the first column of the matrix (*m*11, *m*21, *m*31, *m*41) and the four products are summed, and then the process continues with the second, third and fourth columns.

The vector (*x, y, z, w*) consists of four numbers stored in adjacent memory. Consider hardware that implements parallel processing of matrix multiplication. Do you think it might be faster to perform these four multiplications in parallel if the numbers in each column were also stored in adjacent memory? This seems very likely, which implies the optimum way to store the matrix values in memory is the order *m*11, *m*21, *m*31, *m*41, *m*12, *m*22 and so forth.

> It's certainly possible to define a dynamic vertex buffer that changes at the frame rate of the video display, and in some cases it's necessary.

That's known as column-major order. The memory block begins with the first column of the matrix, then the second, third and fourth. And this is what vertex shaders assume to be the organization of matrices in memory when performing these multiplications.

However, this isn't the way DirectX normally stores matrices in memory. The XMMATRIX and XMFLOAT4X4 structures in the DirectX Math library store matrices in row-major order: *m*11, *m*12, *m*13, *m*14, *m*21, *m*22 and so forth. This seems like a more natural order to many of us because it's similar to the order that we read lines of text—across and then down.

Regardless, there's an incompatibility between DirectX and shader code, and that's why you'll notice in the code in **Figure 3** that every matrix is subjected to an XMMatrixTranspose call before being sent off to the vertex shader. The XMMatrixTranspose function converts row-major matrices to column-major matrices, and back again if you need that.

That's the most common solution to this problem but it's not the only solution. You can alternatively specify a flag to compile the shader for row-major order, or you can leave the matrices untransposed and just switch around the order of the matrix and the vector in the multiplications:

```
pos = mul(viewMatrix, pos);
```

The Final Step

The shading of the three triangles certainly demonstrates that vertex colors are interpolated over the surface of each triangle, but the result is rather crude. Surely I can do better in using this powerful shading tool to mimic the reflection of light. That will be the final step in this plunge into the versatile world of Direct2D. ■

Figure 4 **The Vertex Shader for the Rotating Triangle Effect**

```
// Per-vertex data input to the vertex shader
struct VertexShaderInput
{
  float3 position : MESH_POSITION;
  float3 color : COLOR0;
};

// Per-vertex data output from the vertex shader
struct VertexShaderOutput
{
  float4 clipSpaceOutput : SV_POSITION;
  float4 sceneSpaceOutput : SCENE_POSITION;
  float3 color : COLOR0;
};

// Constant buffer provided by effect.
cbuffer VertexShaderConstantBuffer : register(b1)
{
  float4x4 modelMatrix;
  float4x4 viewMatrix;
  float4x4 projectionMatrix;
};

// Called for each vertex.
VertexShaderOutput main(VertexShaderInput input)
{
  // Output structure
  VertexShaderOutput output;

  // Get the input vertex, and include a W coordinates
  float4 pos = float4(input.position.xyz, 1.0f);

  // Pass through the resultant scene space output value
  output.sceneSpaceOutput = pos;

  // Apply transforms to that vertex
  pos = mul(pos, modelMatrix);
  pos = mul(pos, viewMatrix);
  pos = mul(pos, projectionMatrix);

  // The result is clip space output
  output.clipSpaceOutput = pos;

  // Transfer the color
  output.color = input.color;

  return output;
}
```

Charles Petzold *is a longtime contributor to* MSDN Magazine *and the author of "Programming Windows, 6th Edition" (Microsoft Press, 2013), a book about writing applications for Windows 8. His Web site is charlespetzold.com.*

Thanks to the following Microsoft technical expert for reviewing this article: *Doug Erickson*

DAVID S. PLATT

A Girl's Road to Geekdom

I am 14 years old: a teenager who may one day join your geeky ranks (I'm also Dave's daughter, but I try not to think about that). But before you dismiss me as a kid, consider this: I am your future. And this is how I see you.

We live in the golden age of geeks. Society would be helpless without you behind the scenes, running programs, slinging code and figuring out ways to make devices and software talk to each other. Without you, people would still be hunting and gathering and using Windows XP. The hours might be terrible, but at least you enjoy the respect of the lay folk

I've seen the software industry's efforts to recruit more women in college, and sometimes high school. Let me tell you, that's way too late. We're making up our minds now—in seventh grade or even sixth.

I can assure you that no such respect extends to middle school, where knowledge of who's dating whom matters more than caring about and understanding the nuts and bolts on which our society and world depend. Schools don't hang banners for academic achievements, or throw pep rallies for the coding team.

This irritates me. I have always carried an intense desire to know how things work. When I first started reading, the only text editor software I knew of was NotePad. I wondered how the italics in books were done—handwritten in every copy of the book ever printed? When I was exposed to Microsoft Word, my world suddenly expanded. My earliest questions were answered, but more emerged: How were different-shaped and -sized letters and colors made? When I discovered fonts and colors, I had to change the formatting of every paragraph because it was fun. I began to have the makings of a geek.

Geeks care about how things work—the universe, plants, car engines, atomic structure, anything. As author John Green says, "When people call you a [geek], they're basically saying, 'You like stuff.... You are too enthusiastic about the miracle of human consciousness!'"

But as a teenage girl, the challenge to be taken seriously as a geek is continual and exhausting. It has some advantages, such as intelligent discussions—three of my friends and I spent an entire study hall theorizing about time travel. It also comes with disadvantages, such as hanging out with guys who smell really bad, think they're smarter than you when exactly the opposite is true, undermine your ideas because you're a girl, or all of the above.

I've seen the software industry's efforts to recruit more women in college, and sometimes high school. Let me tell you, that's way too late. We're making up our minds now—in seventh grade or even sixth. My teachers have (too often) expounded that during our middle school years we grow more than any other time of our lives outside of infancy. It is the perfect time to present software as a career, at the moment when we are most malleable.

It wouldn't be hard. Start coding clubs in middle school rather than high school. Have advisers personally reach out to invite girls, and encourage them to bring their friends. Have women from the industry present technical topics that we middle schoolers would find cool. Imagine Parisa Tabriz, Google's self-described "security princess," talking about how to keep your accounts safe from prying parents!

That's why I suggested to Dave at the beginning of the summer that I intern with him to experience what being a full-time geek really entails. He's having me convert his favorite text-based Star Trek game to a Windows Forms program, and then to tablets and phones. (He's such an old fart, but I do have to admit that blowing up Klingons is kind of addicting.) By the time you read this, I'll have some idea if joining the geek ranks is what I want. I suppose a lot will depend on whether either of us is still alive.

Wish me luck, or better yet, wish Dave luck. He's going to need it. ■

ANNABELLE ROSE PLATT *is about to enter eighth grade in Ipswich, Mass. Her world greatly expanded at age 2, when she peeled the tape off her fingers and discovered the numbers 9 and 10. She does not agree with everything her father says.*

Work With Relational Data, Not Complex APIs or Services

Whether you are a developer using ADO.NET, JDBC, OData, or MySQL, or a systems integrator working with SQL Server or Biztalk, or even an information worker familiar with ODBC or Excel – our products give you bi-directional access to live data through easy-to-use technologies that you are already familiar with. If you can connect to a database, then you will already know how to connect to Salesforce, SAP, SharePoint, Dynamics CRM, Google Apps, QuickBooks, and much more!

Give RSSBus a try today and see what mean:

visit us online at www.rssbus.com to learn more or download a free trial.

rssbus

INTEGRATION YOUR WAY

8TH ANNUAL

From the organisers of DEVWEEK

SOFTWARE ARCHITECT

14 – 17 OCTOBER 2014 | HOTEL RUSSELL, LONDON

THE TECHNICAL CONFERENCE FOR SOFTWARE ARCHITECTS AND SOFTWARE DEVELOPERS

BOOK NOW

BOOK YOUR PLACE BY 1 AUGUST AND SAVE UP TO £200

World-renowned speakers, including

ALLEN HOLUB
SIMON BROWN
RUTH MALAN
NEAL FORD

and many more

Featuring

42 BREAKOUT SESSIONS

12 FULL-DAY WORKSHOPS

SOFTWARE ARCHITECT CONFERENCE 2014

software-architect.co.uk

AGILE | ARCHITECTURE | BIG DATA | CLOUD | DATABASE | LEADERSHIP | MOBILE | PATTERNS
PROGRAMMING | PROJECT MANAGEMENT | SECURITY | TEST | UI/UX | WEB

Welcome

We are delighted to announce that the 8th annual Software Architect Conference will be returning to London once again this October.

Software Architect returns in 2014 with more content than ever before. Featuring **12 full-day workshops** and **42 focused 90-min breakout sessions,** our expert speakers will ensure you and your team understand the fundamentals of software architecture and are up to date with the latest developments, ideologies and best practices.

A mixture of practical 'real-world' sessions alongside theoretical overviews and industry perspectives will allow you to walk away with new ideas and approaches on how to perform your job more effectively.

So, if you are – or want to become – a software architect, we hope to see you in October!

The Software Architect Team

2014 SPEAKERS
MEET THE EXPERTS

The 2014 speaker faculty is our most impressive yet! They are acknowledged experts in their field, many with international recognition. We are delighted to confirm that **Allen Holub** and **Neal Ford** will be delivering the keynote presentations this year, while the event also features sessions and workshops from highly-regarded authorities such as **Simon Brown, Eoin Woods, Sander Hoogendoorn, Ruth Malan, Dino Esposito** and many, many more! For a full run down on our 2014 speakers, please refer to pages 14-15.

Top reasons to attend Software Architect 2014

1

12 Full-day workshops

The pre- and post-conference workshops will ensure you increase your knowledge and gain practical advice to take back to the office.

The wide range of workshops, covering topics such as mobile apps, Big Data, BDD, agile and much more, means there is something for everyone. Find out more about the workshops we have available on pages 6-7 and 12-13.

2

Create your own agenda

The main conference features six streamed tracks, covering a wide range of topics. As such, you can create your perfect agenda and make the most of your time at Software Architect.

To help you plan your time, the Software Architect website has an interactive agenda tool, allowing you to select the talks you want to attend and email your personalised agenda to yourself, your colleagues or even your boss – the perfect way to convince your management of why you need to attend!
software-architect.co.uk/agenda

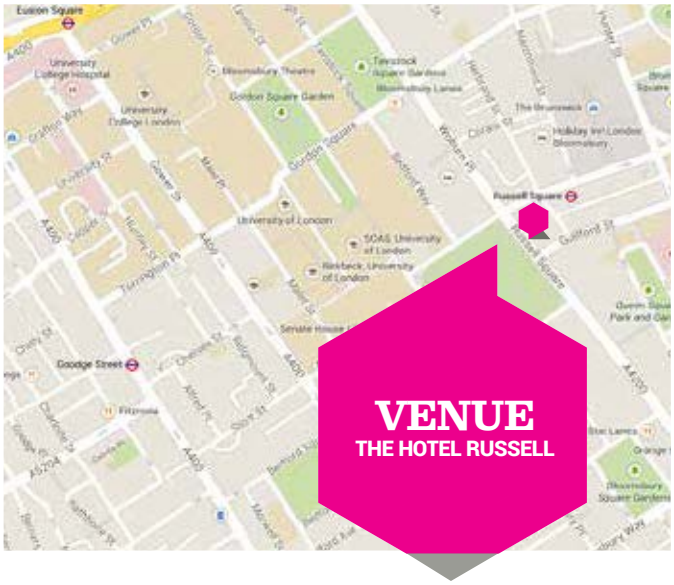
Venue

When visiting the United Kingdom's capital, there is no destination quite like the Hotel Russell. Situated in the very centre of London, in the heart of Bloomsbury, this historic building dominates the east side of Russell Square. Enjoy the peaceful, tranquil greenery of Russell Square Garden, while being close to all the attractions, financial and commercial districts of London. The Hotel Russell is ideally located, being only a 10-minute walk from **Euston, St Pancras** and **Kings Cross stations.** Right next door to the hotel you will find the **Russell Square tube station**, where a short journey on the Piccadilly Line will take you to Covent Garden, Piccadilly Circus and Leicester Square.

For more information, please visit **hotelrussellllondon.co.uk**

Address:
Hotel Russell
1-8 Russell Square
London
WC1B 5BE

Phone:
+44 (0) 2078 376 470



Agenda guide

To help you navigate the wealth of content at Software Architect 2014, all breakout sessions and workshops within this brochure have been colour-coded as a reference to the top-level topic they are covering. Please refer to the agenda guide below. A quick, at-a-glance agenda of the main conference days is available on pages 4-5. The full agenda for all four days, with presentation abstracts, can be found on pages 6-13. For more detailed presentation abstracts, please visit **software-architect.co.uk/agenda**

- **Agile**
- **Architecture**
- **Big Data**
- **Cloud**
- **Database**
- **Leadership/ Project Management**
- **Mobile**
- **Patterns**
- **Programming**
- **Security**
- **Test**
- **UI/UX**
- **Web**

3

Share your ticket

With so much relevant content, we advise that you attend the whole event and bring some colleagues with you to ensure you see as many of the available sessions first-hand.

However, if you only have time to attend one day or others in your team need to attend in your place, you can share your ticket and ensure your company makes the most out of the full four days of sessions and workshops. Full details on how to share your ticket are outlined during the online booking process at **software-architect.co.uk/book**

4

Don't miss a session

Can't be in two places at once? Catch up with any vital breakout sessions you missed, online, after the event.

With six concurrent tracks, there will inevitably be times when you have to choose between two equally important sessions. This year, you won't miss out! All breakout sessions will be filmed and provided online for all registered delegates.

Did you know?

You can highlight your favourite sessions and create your own personalised agenda by using the interactive agenda at software-architect.co.uk/agenda

This **at-a-glance agenda** provides an overview of all the workshops and breakout sessions that are taking place over all four days of the event, allowing you to quickly highlight the key sessions you want to attend.

DAY 1 Pre-conference all-day workshops | Tuesday

Allen Holub
AGILE/DESIGN FROM START TO FINISH
Workshop ref: SA01

Nathaniel Schutta
DESIGNING AND PROTOTYPING MOBILE APPLICATIONS
Workshop ref: SA02

Gary Short
FORECASTING HORSE RACING FOR FUN AND PROFIT
Workshop ref: SA03

Ido Flatow
SECURING ASP.NET APPLICATIONS AND SERVICES: FROM A-Z
Workshop ref: SA04

Ruth Malan
ARCHITECTING: IT'S (NOT) WHAT YOU THINK
Workshop ref: SA05

Seb Rose
BDD BY EXAMPLE
Workshop ref: SA06

DAY 2 Main conference | Wednesday

9.30

WELCOME ADDRESS AND KEYNOTE PRESENTATION

Neal Ford
THIS IS WATER
In this keynote, Neal will describe the water you swim in but cannot see anymore, such as relational databases and application servers. And he will jerk you out of this water (briefly) to describe a strange, fantastical world with things such as immutable database servers and phoenix machines.

Allen Holub
THE DEATH OF AGILE
In this keynote, Allen will look at the state of the agile community as it moves away from the basic principles of agility, at how we got to that state, and at what we need to do to fix things (or get off on the right foot if you're just starting the journey).

11.30

Nathaniel Schutta
MIND THE GAP: ARCHITECTING UIS IN THE ERA OF DIVERSE DEVICES

Eoin Woods
SYSTEM SECURITY BEYOND THE LIBRARIES

Sander Hoogendoorn
SHAPING SERVICE ORIENTATION WITH SMART USE CASES

Austin Bingham
OPEN DECISIONS IN ARCHITECTURAL EVOLUTION

Jules May
IF CONSIDERED HARMFUL: HOW TO ERADICATE 95% OF ALL YOUR BUGS IN ONE SIMPLE STEP

Michael Kennedy
HIGH-PERFORMANCE NOSQL TECHNIQUES IN .NET

14.00

Sasha Goldshtein
MODERN BACKENDS FOR MOBILE APPS

Ed Courtenay
AN INTRODUCTION TO TYPESCRIPT

Neal Ford
EMERGENT DESIGN

Howard Deiner
LEAN THINKING AND WHAT IT MEANS TO THE AGILE MINDSET

Seb Rose
MUTATION TESTING – BETTER CODE BY MAKING BUGS

Gary Short
FROM ZERO TO HADOOP

16.00

Dino Esposito
THE SKETCHY THING AND THE COMEBACK OF FLOWCHARTS AND TOP-DOWN DESIGN

Ido Flatow
BUILDING SCALABLE, DURABLE AND SECURED WEB APPLICATIONS WITH THE MICROSOFT AZURE PLATFORM

Nuno Filipe Godinho
IOT & M2M: HOW ARE THEY CHANGING THE WORLD WE LIVE IN?

Brian A. Randell
BUILDING A DEVOPS CULTURE

Allen Holub
TDD, BDD & ATDD: TEST-, BEHAVIOUR- AND ACCEPTANCE-TEST-DRIVEN DEVELOPMENT

Jules May
BIG DATA AND PRIVACY

ALSO EACH DAY: 8.30 Coffee & registration | 11.00 Coffee break | 13.00 Lunch break | 15.30 Coffee break

DAY 3 Main conference | Thursday

9.30	11.30	14.00	16.00
Jules May MOBILISING WEBSITES	**Sasha Goldshtein** C# EVERYWHERE: CROSS-PLATFORM APPS WITH XAMARIN	**Sasha Goldshtein** SWIFT: APPLE'S NEW PROGRAMMING LANGUAGE FOR IOS AND OS X	**Nathaniel Schutta** BUILDING A MOBILE COMPETENCY CENTRE
Shay Friedman ANGULARJS: THE ONE FRAMEWORK TO RULE THEM ALL	**Nuno Filipe Godinho** ARCHITECTURE BEST PRACTICES ON WINDOWS AZURE	**Brian A. Randell** APPLICATION ANALYTICS – ADD THE RIGHT FEATURES TO YOUR APP	**Allen Holub** WEB APPLICATION ARCHITECTURE: THE WHOLE STACK
Simon Brown & Eoin Woods MODELS, SKETCHES AND EVERYTHING IN BETWEEN	**Brian A. Randell** VISUAL STUDIO ONLINE: AN UPDATE EVERY THREE WEEKS	**Neal Ford** CONTINUOUS DELIVERY FOR ARCHITECTS	**Ralf Westphal** LET SOFTWARE DESIGN COME TO LIFE USING SOFTWARE CELLS
Robert Smallshire CONDUCTING A SOFTWARE ARCHITECTURE REVIEW	**Sander Hoogendoorn** INDIVIDUALS AND INTERACTIONS OVER PROCE$$E$ AND FOOLS	**Seb Rose** LIES, DAMN LIES AND ESTIMATES	**Howard Deiner** AGILE/LEAN STARTUP IT PORTFOLIO MANAGEMENT – NO LONGER OXYMORPHIC!
Ed Courtenay WHAT ARE AUTO-MOCKING CONTAINERS AND WHY SHOULD YOU USE THEM?	**Dino Esposito** DDD MISCONCEPTIONS	**Allen Holub** DESIGN PATTERNS IN THE REAL WORLD	**Gil Fink** QUICK TOUR TO FRONT-END TESTING USING JASMINE
János Fehér CREATING ADAPTIVE ARCHITECTURE WITH ZEROMQ	**Gary Short** BEST PRACTICE ARCHITECTURE IN HETEROGENEOUS BIG DATA STORES	**Nuno Filipe Godinho** BIG DATA IN THE ENTERPRISE	**Michael Kennedy** APPLIED NOSQL WITH MONGODB AND PYTHON

DAY 4 Post-conference all-day workshops | Friday

Neal Ford CONTINUOUS DELIVERY Workshop ref: SA07	**Gil Fink & Ido Flatow** END-TO-END SPA DEVELOPMENT Workshop ref: SA08	**Ralf Westphal** LIGHTWEIGHT AGILE SOFTWARE ARCHITECTURE FOR TEAMS Workshop ref: SA09
Andy Clymer & Richard Blewett A DAY OF DESIGN PATTERNS AND TESTING Workshop ref: SA10	**Paul Ardeleanu** BUILDING DATA-DRIVEN MOBILE APPS USING AMAZON WEB SERVICES Workshop ref: SA11	**Seb Rose** APPLIED BDD WITH CUCUMBER, CUCUMBER-JVM AND SPECFLOW Workshop ref: SA12

DAY 1 Pre-conference all-day workshops | Tuesday

The following workshops run for a **full day, from 09:30 to 17:30** with a short break in the morning and afternoon, and a lunch break at 13:00. You either require a one-day workshop pass to attend **OR** you can book a 3-day Pass or Universal Pass and attend the main conference sessions as well.

Tuesday 14 October

AGILE/DESIGN FROM START TO FINISH

Allen Holub

Agile systems are, by necessity, tightly coupled to the user's notion of what the system is doing. Without that connection, the software can't stand up to the stress of constant change that all agile processes mandate. Moreover, the process that you use influences the architecture of your system. That's why hybrid processes that attempt to mix agile and traditional practices often fail. The hybrid architectures that come out of those processes are unworkable.

In this workshop, Allen talks about both process and architecture. We'll look at how agile processes work, and show how both architecture and low-level design naturally fall out of those processes. Specifically, we'll look closely at the role of stories, at how they're created and developed, and how they flow through the process, with a focus on how to develop an optimal architecture that closely mirrors both the stories themselves and the assumptions that underlie the stories.

We'll also work through a real-world example that demonstrates the process from front to back: requirements gathering and problem-statement definition, story development (use-case analysis), and the simultaneous construction of the light-weight dynamic and static models that underlie the code.

Workshop ref: SA01

DESIGNING AND PROTOTYPING MOBILE APPLICATIONS

Nathaniel Schutta

The word just came down from the VP: you need a mobile app and you need it yesterday. Wait, you've never built a mobile app... It's pretty much the same thing as you've built before, just smaller, right? Wrong. The mobile experience is different and far less forgiving. How do you design an application for touch? How does that differ from a mouse? Should you build a mobile app or a mobile web site?

In this workshop, Nathaniel will get you started designing for a new and exciting platform. Whether that means iPhone, Android, Windows Phone or something else, you need a plan – and this workshop will help.

We'll start by discussing what it means to design within the constraints of mobile devices. Using a widely used web site, we'll drive through the design process, creating personas and talking about what capabilities they would need in a mobile context.

In the second part of this all-day workshop, we'll leverage jQuery Mobile to prototype the designs we created in the morning. Even if you decide to build a native application, jQuery Mobile is an invaluable tool; using HTML5 and simple conventions, you can quickly and painlessly build interactive web sites.

Workshop ref: SA02

FORECASTING HORSE RACING FOR FUN AND PROFIT

Gary Short

Data Science and Big Data are both hot topics at the moment, and while it is true that data science can be hard, it can also be fun! Trying to learn the techniques in classical and Bayesian statistics from abstract example in textbooks and web pages can be a real challenge, and when you throw in Machine Learning, it can become almost impossible to fathom.

In this workshop, Gary will use the medium of predicting horse racing results in order to anchor your learning in something concrete (and fun). Throughout the day, we'll explore and implement a number of regression models from classic statistics, before going on to look at the impact Bayesian Inference can have on the problem. After lunch, we'll throw a few Machine Learning techniques into the mix, to ensure you have a view of the problem using all the "data science toys".

Lastly, we'll round out the day by combining a few of these into a powerful "mixed model" prediction engine, learning how to weight the output from each model, thus creating a powerful prediction engine that we can use to predict horse racing results.

Workshop ref: SA03

> Great range of speakers & topics, giving plenty of food for thought and tools to take back to work.
> TECHNICAL ARCHITECT

BETWEEN SESSIONS: 8.30 Coffee & registration | 11.00 Coffee break | 13.00 Lunch break | 15.30 Coffee break

SECURING ASP.NET APPLICATIONS AND SERVICES: FROM A-Z

Ido Flatow

When you think of ASP.NET security, the first thing that comes to mind are Windows authentication and Forms authentication using ASP.NET Membership. For years, those were the common authentication techniques for ASP.NET applications and services. But with the new releases to the ASP.NET Identity system, those days are long gone. For the enterprise, ASP.NET broadened its support from the on-premises Active Directory to include Microsoft Azure Active Directory. By supporting external identity providers, such as Facebook, Microsoft Account and Twitter, the new ASP.NET Identity system makes the process of securing an application less scary than ever.

In this workshop, Ido will start from the basics: getting to know concepts such as SSL, OAuth, OpenID and claim-based authorisation. From there we will continue to explore the various scenarios of using self-managed identities, Active Directory and ADFS, external identity providers (Facebook, Google, Microsoft), and Microsoft Azure Active Directory.

Workshop ref: SA04

ARCHITECTING: IT'S (NOT) WHAT YOU THINK

Ruth Malan

The arc of software architecture encompasses various decisions and considerations, resolving important sets of interacting tensions. Among them:

- **Strategic and structural significance:** identifying strategic outcomes and defining challenges; design of system capabilities and system structure; system qualities and mechanism design
- **Decision scope:** decisions at broader scopes (system, mechanism, service) and decisions at narrow or local scopes (units), considering intentionality and emergence
- **Timing of decisions:** clearing the fog of uncertainty/putting ground under the feet and the "last responsible moment", iteration and evolution

In this workshop, we will spend time with the usual suspects – (re)factoring, dependencies, naming, forces, trade-offs, mechanism design, system and component boundaries and interaction surfaces. And some sketchy ones – making the system design visual and drawing people in. We will take some silver bullets – relationships of goodwill and commitment to objectivity – to heart, and be playful. And we will take our fallibilities, biases and foibles into account.

How? That is indeed it. Our focus will be on how. We will create a draft (set of views of the) software architecture to situate our discussions and practice system thinking and modelling, strategic thinking (understanding what is shapingly important in the user context and business and technology space), and design improvement strategies. Our orientation is the co-creation of systems that have desired structural integrity properties, including resilience, but also design integrity and dynamic unity.

Our goal is to surface key matters of architectural judgment, drawing out myths and misconceptions, and sharing, positioning and connecting useful conceptions, strategies and techniques, and laws, principles, heuristics and other guidance.

Workshop ref: SA05

BDD BY EXAMPLE

Seb Rose

In this workshop, Seb will provide a practical introduction to using examples to specify software. You will learn to break down complex business requirements with your stakeholders, using examples in their own language, giving you the tools you need to explore their ideas before you even write any software.

This workshop is for everybody involved in the process of developing software, so please bring product owners, testers and architects along. As well as describing what BDD is (and isn't), we'll spend a lot of time practicing collaborative analysis to make sure that our stories are appropriately sized, easy to read and unambiguous. We'll develop a "ubiquitous language", explore the workings of the Three-Amigos meeting, and really get to grips with the slippery interaction between features, stories, acceptance criteria and examples.

Before the day is over, we'll also take time to discuss:

- How BDD relates to SBE, ATDD and TDD
- Approaches to the cost/benefit trade-off of test automation
- Where BDD fits into your existing development process
- The benefits of "living documentation"

We'll use pens, cards and other bits of paper, so you won't need to know any tools in advance, or even remember your laptop!

To build further on these ideas, don't miss Seb's post-conference workshop: "Applied BDD with Cucumber, Cucumber-JVM and SpecFlow".

Workshop ref: SA06

DAY 2 Main conference | Wednesday

Take a closer look at the abstracts for all the sessions during the main conference on Wednesday, 15 October. For more detailed abstracts, please visit: **software-architect.co.uk/agenda**

Wednesday 15 October

9.30

WELCOME ADDRESS AND KEYNOTE PRESENTATIONS

Neal Ford

THIS IS WATER

A fish is swimming along, having some breakfast, when suddenly he's snatched out of his world by his food; abruptly ejected into a bright world where it's hard to breathe; landing on the bottom of a boat, where strange alien creatures make strange sounds. Overhead, he sees an airplane flying at 500mph. Suddenly, one of the creatures picks him up, removes the hook, and, just as suddenly, he's back home. Yet, when he regales his friends with this tale, no one can believe such a strange world can exist.

In this keynote, Neal will describe the water you swim in but cannot see anymore, such as relational databases and application servers. And he will jerk you out of this water (briefly) to describe a strange, fantastical world with things such as immutable database servers and phoenix machines.

You may have trouble getting your friends who didn't attend to understand.

Allen Holub

THE DEATH OF AGILE

At the very top of the Agile Manifesto is the statement: "Individuals and interactions over processes and tools". In spite of that thought (and its prominence), the word "agile" has come more and more to mean rigid adherence to specific processes and the tools that support those processes. Moreover, many of these processes are incomplete – Scrum, for example, uses only two of the 13 practices that make up XP – and process adherents rarely add in the missing pieces. The word "agile", however, certainly applies to the process itself.

In this keynote, Allen will look at the state of the agile community as it moves away from the basic principles of agility, at how we got to that state, and at what we need to do to fix things (or get off on the right foot if you're just starting the journey).

11.30

Nathaniel Schutta

MIND THE GAP: ARCHITECTING UIS IN THE ERA OF DIVERSE DEVICES Architecting and developing user interfaces used to be relatively easy: pick a server-side framework, define a standard monitor resolution, and spend your days dealing with browser quirks. But today, the landscape presents us with a plethora of screen sizes and resolutions, covering everything from a phone to a tablet to a TV. How does a team embrace this brave new world, knowing that the future will introduce even more volatility to the client space? ◇◆

Eoin Woods

SYSTEM SECURITY BEYOND THE LIBRARIES Security is now important to all of us, not just people who work at Facebook. But it is a complicated domain, with a lot of concepts to understand. In any technical ecosystem, there is a blizzard of security technology, as well as generic concepts such as keys, roles, certificates, trust, signing and so on. Yet, none of this is useful unless we know what problem we're really trying to solve. In this session, Eoin will dive into the fundamentals of system security in order to decide how to secure our systems. ◆

Sander Hoogendoorn

SHAPING SERVICE ORIENTATION WITH SMART USE CASES Although many organisations apply service-oriented architecture at the core of their software development efforts, executing such projects is hard. There are many different stakeholders, even external to the organisation. And then there are many different types of deliverables. In this session, Sander will walk you through the difficulties that surround service-oriented projects, but will also demonstrate the use of a particularly useful modelling technique, called "smart use cases". ◇

Austin Bingham

OPEN DECISIONS IN ARCHITECTURAL EVOLUTION Evolving software architectures involves balancing many factors, and maintaining that balance can be challenging for any architect. By opening up the decision process for evolution, we can harness the insight of fellow developers, communicate plans and designs more effectively, and produce a useful record of the work we do. In this session, Austin will look at a specific lightweight technique – "Open Design Proposals" – which has proven its effectiveness in many projects. ◇◆

Jules May

IF CONSIDERED HARMFUL: HOW TO ERADICATE 95% OF ALL YOUR BUGS IN ONE SIMPLE STEP In 1968, CACM published a letter from Edgar Dijkstra, called "Go To statement considered harmful". In it, he explained exactly why most bugs in programs were caused by Gotos, and he appealed for Goto to be expunged from programming languages. But Goto has a twin, which is responsible for nearly every bug that appears in our programs today. That twin is "If". In this session, Jules will show why If and Goto have the same pathology. ◆

Michael Kennedy

HIGH-PERFORMANCE NOSQL TECHNIQUES IN .NET You're one of the brave ones who has jumped into the NoSQL pool and found it a refreshing change. That's awesome. But there is so much more to being successful with NoSQL databases than simply getting started. In this session, Michael will explore some of the issues, techniques and best practices for being successful with NoSQL, in general, and MongoDB, in particular. This includes exploring correct document/entity design, indexes and deployment – to name just a few of the topics. ◆

BETWEEN SESSIONS: 8.30 Coffee & registration | 11.00 Coffee break | 13.00 Lunch break | 15.30 Coffee break

14.00

Sasha Goldshtein

MODERN BACKENDS FOR MOBILE APPS In this session, Sasha will explore cloud backend providers that take away the burden of managing servers and infrastructure for your mobile apps. Backend concerns for mobile apps include data storage, efficient queries, push notifications and user authentication. We will explore the system architecture and concrete design for mobile apps that rely on Azure Mobile Services, Facebook Parse and cloud storage providers as a ready-made backend.

Ed Courtenay

AN INTRODUCTION TO TYPESCRIPT JavaScript is the scripting glue that holds the web together – largely because of its flexibility. This flexibility also means that it can be difficult to manage, especially in large-scale applications. In this session, Ed will demonstrate how TypeScript can help to tame new and existing client-side code. Using real-world examples, we will explore some of the problems with client-side JavaScript development as a motivating example, and introduce TypeScript as a way of solving some of these issues.

Neal Ford

EMERGENT DESIGN The hazard of Big Design Up Front in software is that you don't yet know what you don't know, and design decisions made too early are just speculations without facts. Emergent design techniques allow you to wait until the last responsible moment to make design decisions. In this session, Neal will cover four areas: emergent design enablers, battling things that make emergent design hard, finding idiomatic patterns, and how to leverage the patterns you find.

Howard Deiner

LEAN THINKING AND WHAT IT MEANS TO THE AGILE MINDSET Long before the agile revolution, industry had learned that efficient production of goods required intense attention to quality, teamwork and continuous improvement. These themes of lean manufacturing were never part of the original formulation of the Agile Manifesto, and are rarely mentioned as part of the traditional agile/scrum recipe. In this session, Howard will teach the basics of lean, and demonstrate how they apply to agile development.

Seb Rose

MUTATION TESTING – BETTER CODE BY MAKING BUGS Do you know how good your tests are? Mutation testing can tell you. Unlike test coverage metrics, mutation testing lets us say something concrete about the quality of your test suite. Mutation testing has been around for years, but it's only recently that performance tools (such as PITest for Java) have become available. In this session, Seb will look at the motivation and technology behind mutation testing and see some examples in action.

Gary Short

FROM ZERO TO HADOOP In this session, Gary will show you how to get started with Hadoop on a Microsoft platform. We'll cover installation and exploration of the environment, before going on to learn how to write map reduce functions in C#, Pig and Hive.

16.00

Dino Esposito

THE SKETCHY THING AND THE COMEBACK OF FLOWCHARTS AND TOP-DOWN DESIGN Always neglected in favour of domain analysis and modelling, the presentation layer of most applications hardly receives due attention. Meanwhile, a new generation of users is emerging that is much less forgiving, while also dictating UI constraints to architects and developers. In this session, Dino discusses a design approach that starts from requirements and builds the system from top to bottom.

Ido Flatow

BUILDING SCALABLE, DURABLE AND SECURED WEB APPLICATIONS WITH THE MICROSOFT AZURE PLATFORM We always hear tips about the patterns and practices of building web applications with Microsoft Azure. It's time we see how it is done! In this session, Ido will construct a secured, durable, scalable, low-latency web application with Microsoft Azure – including hosting, storage, database, cache and Active Directory.

Nuno Filipe Godinho

IOT & M2M: HOW ARE THEY CHANGING THE WORLD WE LIVE IN? Internet of Things (IoT) is here and every day a new sensor or device starts to generate more data. With that, more and more Machine-to-Machine (M2M) communications occur, which means our solutions have new issues to face. In this session, Nuno will look at how these new "buzz words" are changing the world we live in, from fitbit to Google Glass and smart watches. How can we prepare for this? How can we get some business opportunities from it?

Brian A. Randell

BUILDING A DEVOPS CULTURE At the end of the day, development and operations departments exist to delight customers and help the business achieve its goals. The promise of DevOps is to create a closed-loop process, where traditional silos are replaced with collaborative teams that can respond to issues – both technological and business – with agility. In this session, Brian will show how an emphasis on people with better process and some tools can help you achieve a culture of DevOps within your organisation.

Allen Holub

TDD, BDD & ATDD: TEST-, BEHAVIOUR- AND ACCEPTANCE-TEST-DRIVEN DEVELOPMENT TDD (and its specialisations: BDD and ATDD) is a dynamic design process particularly suited for agile environments. That is, (T/B/AT)DD is really a design, not a testing strategy. But it's a design strategy that produces a set of very useful tests as a by-product. By building acceptance tests around tentative interfaces, then incrementally refining those interfaces using tests, you both design and code simultaneously in a way that yields an optimal design.

Jules May

BIG DATA AND PRIVACY We have been talking about Big Data for several years now, but Care.data and Edward Snowden have brought home to the non-technical public just what Big Data really implies – both the benefits and the risks. In doing so, it has opened up a whole new debate that is going to concern not just Big Data processors, but anyone who handles any kind of personal data. Our notion of privacy is about to change, and new regulation is not far behind. If you handle data, even anonymous data, things are about to get interesting.

DAY 3 Main conference | Thursday

Take a closer look at the abstracts for all the sessions during the main conference on Thursday, 16 October. For more detailed abstracts, please visit: **software-architect.co.uk/agenda**

Thursday 16 October

9.30

Jules May

MOBILISING WEBSITES The major challenge facing today's web site designers is adapting content to a range of distinct clients. Mobile and tablet versions are in the news just now, but there's also integration with on-device apps, interfaces for disabled users, rights management and white-labelling. In this session, Jules will describe how he built an "impedance-matching" function at the server, downstream of the application, performing technical negotiation with the client, and delivering multiple user experiences from a single application.

Shay Friedman

ANGULARJS: THE ONE FRAMEWORK TO RULE THEM ALL In the last couple of years we've seen the rise of client-side JavaScript frameworks. From almost nothing, we now have at least a dozen to choose from. One of the new kids in the block, AngularJS, comes straight from the Google offices and tries to stand out from the crowd with a complete set of tools and utilities. In this session, Shay will go through the different features of AngularJS and see what makes it one of the most popular JavaScript frameworks out there.

Simon Brown & Eoin Woods

MODELS, SKETCHES AND EVERYTHING IN BETWEEN Just the mention of the word "modelling" brings back horrible memories of analysis paralysis for many software developers. And, in their haste to adopt agile approaches, we've seen countless software teams who have thrown out the modelling baby with the process bathwater. In extreme cases, this has led to the creation of software systems that really are the stereotypical "big ball of mud". In this session, Simon and Eoin will discuss models, sketches and everything in between.

Robert Smallshire

CONDUCTING A SOFTWARE ARCHITECTURE REVIEW A project struggling to deliver the right balance of qualities; due diligence in advance of an acquisition; an independent assessment of software you have commissioned – these are all good motivations for a software architecture review – a candid examination of how a software design resolves the forces acting upon it. But how to conduct an architecture review? In this session, Robert will define goals for a review, explain who needs to be involved, when and why, and discuss key topics you should cover.

Ed Courtenay

WHAT ARE AUTO-MOCKING CONTAINERS & WHY SHOULD YOU USE THEM? Explore how to use auto-mocking containers to improve and streamline your unit tests, and also how to exercise your IoC container in a testable fashion. In this session, Ed will cover lessons learned from a large development project that can help you avoid the same pitfalls in your own code. Starting from a simple project, the fragility of existing unit tests will be demonstrated, and we will see how developers can be discouraged from making comprehensive tests.

János Fehér

CREATING ADAPTIVE ARCHITECTURE WITH ZEROMQ In this session, János will demonstrate how architects and lead engineers can design a flexible but lightweight distributed architecture with ZeroMQ. At Intern Avenue, with this pattern, his team were able to glue together different frameworks and programming languages. When they separated components, the architecture was flexible enough to scale without changing the code from inter-process communication, to Unix socket-based to a clustered solution.

11.30

Sasha Goldshtein

C# EVERYWHERE: CROSS-PLATFORM APPS WITH XAMARIN Are you a C# developer? Do you need to develop an iOS, Android, Windows Phone app? You can write your app in C# and share up to 90% of the code using Xamarin's tools. In this session, Sasha will develop a cross-platform C# app that uses Xamarin to run on iOS, Android and Windows Phone. We'll talk about developing views for each platform, sharing business logic and accessing HTTP services. We will also discuss options for introducing a cross-platform MVVM design into your app.

Nuno Filipe Godinho

ARCHITECTURE BEST PRACTICES ON WINDOWS AZURE When new technologies and paradigms appear, it is essential to learn them quickly and well. But this can be very hard, since some things are only learned with experience. In this session, Nuno will look at some architecture best practices that will help us make our solutions better across several levels, including performance, cost, integration, security and so on. By doing this, you'll gain the knowledge needed to quickly start using the technology and paradigms that can improve your business.

Brian A. Randell

VISUAL STUDIO ONLINE: AN UPDATE EVERY THREE WEEKS If you're looking for a no-hassle way to manage your development team's assets using centralised version control or Git; if you want to track your work on projects big or small; if you want to have elastic build servers on demand; and even monitor your apps for availability; then Brian's insights into Visual Studio Online might be for you. Whether you build Windows apps, OS/X apps, mobile apps on iOS or Android – and even Linux – you'll have something to see.

Sander Hoogendoorn

INDIVIDUALS AND INTERACTIONS OVER PROCE$$E$ AND FOOLS The first statement in the Agile Manifesto clearly favours individuals, teams, interaction and collaboration over processes and tools. But there are two sides to every story. When it comes to tools, the Agile Manifesto is often misinterpreted in that it is not allowed to use tooling in agile projects. In this session, Sander will shine his critical light on the sense and nonsense of tools in the agile field, from post-its and brown-paper to wikis, agile dashboards and Scrumdamentalism.

Dino Esposito

DDD MISCONCEPTIONS For too long, domain-driven design (DDD) has been sold as the ideal solution for very complex problems that only a few teams are actually facing. While technically correct, this statement sparked a number of misconceptions. In this session, Dino will clear the ground around DDD, emphasising the theoretical pillars of the approach: ubiquitous language and bounded context. From there, he'll move ahead to creating a context map and then finally touch on the most commonly used supporting architectures for DDD.

Gary Short

BEST PRACTICE ARCHITECTURE IN HETEROGENEOUS BIG DATA STORES In this session, Gary will cover best practice architectures for heterogeneous Big Data stores, covering SQL, NoSQL and distributed systems, and how best to make them interoperable.

BETWEEN SESSIONS: 8.30 Coffee & registration | 11.00 Coffee break | 13.00 Lunch break | 15.30 Coffee break

14.00

Sasha Goldshtein

SWIFT: APPLE'S NEW PROGRAMMING LANGUAGE FOR IOS AND OS X
At WWDC 2014, Apple announced Swift, a new programming language for iOS and OS X. In this session, Sasha will review Swift's fundamental concepts, including built-in types and collections, optionals, closures, protocols, extensions, generics and custom operators. We will see how Swift improves on Objective-C in terms of type safety, readability and developer productivity, and demonstrate Swift's capabilities by building a simple iOS application from the ground up.

Brian A. Randell

APPLICATION ANALYTICS – ADD THE RIGHT FEATURES TO YOUR APP
You've shipped your first app. Now what? Your backlog is a mile long with wishes from users, as well as your great ideas. But what should you implement? In this session, Brian will first introduce the what and the how of application analytics. He'll then dig into options you can use, both free and paid, to instrument your cross-platform solutions, giving you the right information from your users, so that you can build the next best release of your app.

Neal Ford

CONTINUOUS DELIVERY FOR ARCHITECTS Continuous delivery is a process for automating the production readiness of your application every time a change occurs – to code, infrastructure or configuration. In this session, Neal takes a look at the intersection of the architect role and the engineering practices in continuous delivery. In the continuous delivery world, rather than hone skills at predicting the future via Big Design Up Front, the emphasis lies with techniques for understanding and changing code with less cost during the process.

Seb Rose

LIES, DAMN LIES AND ESTIMATES Are estimates an essential part of project planning and delivery, or a waste of everybody's time? As is so often the case, the answer is neither and both. In this session, Seb will demonstrate that there is more than one kind of estimate, and examine how they are typically used in an agile context. In doing so, we will look at what some of the great minds have said on the subject, from Disraeli to Steve McConnell. Ultimately, Seb may not change your mind, but he intends to widen your perspective.

Allen Holub

DESIGN PATTERNS IN THE REAL WORLD
Design patterns do not exist in splendid isolation. In the real world, patterns overlap one another and interact in complex ways. In this session, Allen takes a unique approach to teaching patterns by analysing a real computer program in terms of the patterns used to implement it. We'll look at how the design patterns are actually used in practice, and see how the strengths and weaknesses of the patterns play off one another.

Nuno Filipe Godinho

BIG DATA IN THE ENTERPRISE Big Data is a popular topic at this point. However, when we talk about large enterprises, we see that a lot of them are still struggling; trying to understand what data makes sense. How can it be integrated with the current IT infrastructure? How to solve the data ingestion problem? In this session, Nuno will be looking at techniques based on real-world experiences that can be used by enterprises to solve these issues in a way that will allow them to take control of Big Data scenarios without losing control of their data.

16.00

Nathaniel Schutta

BUILDING A MOBILE COMPETENCY CENTRE
By now, the importance of having a mobile solution is obvious to just about every seat in the organisation. But how do we develop expertise? How do we work through the inevitable politics and organisational issues? What about the technical questions about hybrid vs. web vs. native? In this session, Nathaniel will explore these obstacles and present an approach to achieving mobile competency within your organisation.

Allen Holub

WEB APPLICATION ARCHITECTURE: THE WHOLE STACK
In this session, Allen will provide an integrated look at the architecture of an entire "vertical slice" of the web-application stack, from the UI at the top to the database at the bottom, and all of the interior plumbing (messaging, caching, etc) in the middle. We'll discuss various trade-offs and alternatives (several UI architectures are available, for example), as well as specific technologies that you can use to implement various components.

Ralf Westphal

LET SOFTWARE DESIGN COME TO LIFE USING SOFTWARE CELLS
Layered and N-tier architecture has reached its limits. Today's technologies for distributed software need a more flexible meta-model for coarse-grained software design. That's what software cells provide. They let you view software as an evolving whole, consisting of autonomous units on multiple levels of detail. Through vertical and horizontal cell division they allow you to think of software as a nested self-similar structure, with which to tackle any size of requirements.

Howard Deiner

AGILE/LEAN STARTUP IT PORTFOLIO MANAGEMENT – NO LONGER OXYMORPHIC! Centralised development resources are common in many of the late adopter organisations that struggle with agile transformations. In this session, Howard focuses on the mechanics of a portfolio management technique intended to guide the organisation into the use of lean startup thinking (with which to challenge product managers), paired with the right type of fiscal rigour (to make CFOs happy).

Gil Fink

QUICK TOUR TO FRONT-END TESTING USING JASMINE
Unit testing is an inseparable part of the development process, no matter which methodology you follow. But how will you test your JavaScript code? The answer is using JavaScript unit testing framework. Jasmine is a behaviour-driven development (BDD) framework for testing JavaScript code. In this session, Gil will introduce Jasmine and you will learn the building blocks of writing your unit tests using the framework.

Michael Kennedy

APPLIED NOSQL WITH MONGODB AND PYTHON NoSQL is a hot topic in the tech industry today. But what exactly is NoSQL and should I use it to build my next application? In this session, Michael will dig into why NoSQL databases are sweeping the industry and discuss the trade-offs between the various types (keyvalue stores vs. document databases, for example). We will explore the most broadly applicable variant of NoSQL, document databases, through hands-on demos with the most popular of the document databases, MongoDB.

DAY 4 Post-conference all-day workshops | Friday

The following workshops run for a **full day, from 09:30 to 17:30** with a short break in the morning and afternoon, and a lunch break at 13:00. You either require a one-day workshop pass to attend **OR** you can book a 3-day Pass or Universal Pass and attend the main conference sessions as well.

Friday 17 October

CONTINUOUS DELIVERY

Neal Ford

Getting software released to users is often a painful, risky and time-consuming process. In this workshop, Neal will set out the principles and technical practices that enable rapid, incremental delivery of high-quality, valuable new functionality to users. Through automation of the build, deployment and testing process, and improved collaboration between developers, testers and operations, delivery teams can get changes released in a matter of hours – even minutes – no matter what the size of a project or the complexity of its code base.

We'll begin by describing the technical differences between related topics such as continuous integration, continuous deployment and continuous delivery, and describe the new technical artefact that continuous delivery introduces, the deployment pipeline. Neal will discuss the various stages, how triggering works, and how to pragmatically determine what "production ready" means

We'll then cover the role of testing and the testing quadrant, including the audience and engineering practices around different types of tests. This is followed by version control usage and offering alternatives to feature branching, such as toggle and branch by abstraction. Neal will then go on to cover operation, DevOps and programmatic control of infrastructure, using tools such as Puppet and Chef. He will also discuss how to incorporate databases and DBAs into the continuous integration and continuous delivery process.

Continuous delivery is a process for automating the production readiness of your application every time a change occurs to code, infrastructure or configuration. It turns out that some architectures and practices yield code that works better in this environment. This workshop takes a deep dive into the intersection of the architect role and the engineering practices in continuous delivery.

Workshop ref: SA07

END-TO-END SPA DEVELOPMENT

Gil Fink & Ido Flatow

Single Page Applications (SPAs) are web applications that are built using a single page, which acts as a shell to all the other web pages, with a rich JavaScript front-end. As opposed to traditional web applications, most of the SPA development is done on the front-end. The server, which once acted as a rendering engine, provides only a service layer to the SPA. This transition helps to create a more fluid user experience (UX), and turns out web applications to desktop-like smart clients.

Known web applications, such as Gmail and Google Docs, are implemented as SPAs, demonstrating that SPAs are the way to write your future web applications and not just a transient trend.

In this workshop, Gil and Ido will demonstrate the concepts of building an end-to-end SPA using an ASP.NET Web API back-end and Backbone.js in the front-end. We will start by learning crucial concepts, such as object-oriented JavaScript and modular JavaScript (patterns and Require.js), that are relevant to SPA development understanding. After setting the background needed for SPA, we will drill down into Backbone.js and Underscore.js which are the libraries we will use in the workshop for MVW and template engine. Later on, you will learn how to create back ends using ASP.NET Web API. And, at the end of the day, you will have a thorough understanding of SPA building blocks and how to create your own SPA.

Workshop ref: SA08

LIGHTWEIGHT AGILE SOFTWARE ARCHITECTURE FOR TEAMS

Ralf Westphal

What makes software architecture agile? It's not SOLID principles or TDD, but progressing incrementally. Increments thus need to become tangible parts of an architecture; tangible for customers as well as for developers. Also – since agility is about people – designing software must become easy to do for the whole team, from requirements down to single functions. Without the ability to think as a team about software on a conceptual level there will be no collective architecture ownership.

In this workshop, Ralf will demonstrate how to approach any requirements in a systematic way and transform them into a series of thinly sliced designs. Those designs can then be jointly implemented in a distributed fashion by a team for fastest feedback. Think feature team instead of feature developer.

Topics covered in this workshop:

- Context-driven software design
- Slicing requirements for incremental delivery
- Finding entry points for functional design
- Two principles for evolvable functional hierarchies
- Three basic categories of responsibilities of code
- Class follows function – how to wrap functionality into containers of various granularity for evolvability and production efficiency
- Rapid delivery through division of labour – components and μServices to the rescue

Workshop ref: SA09

> SA has been delightful; challenging to 180°. All of the sessions were info
>
> CHIEF SOFTWARE ARCHITECT

BETWEEN SESSIONS: 8.30 Coffee & registration | 11.00 Coffee break | 13.00 Lunch break | 15.30 Coffee break

A DAY OF DESIGN PATTERNS AND TESTING

Andy Clymer & Richard Blewett

Re-use solutions, not just code. Design patterns help you identify problems that occur, repeatedly, in your code and solve those problems in a standardised way. The principles that design patterns introduce are also those that make your code testable. The combination of test-driven development and design patterns allows you to make sure that your code is functionally correct and has a robust design.

Design patterns also give you useful tools to help test potentially complex code. In this workshop, Andy and Richard will in no way cover all OO design patterns, but will immerse you in the world of loose coupling and TDD, and set you in good stead to continue your learning, resulting in you building testable object-oriented solutions.

Workshop ref: SA10

BUILDING DATA-DRIVEN MOBILE APPS USING AMAZON WEB SERVICES

Paul Ardeleanu

In the age of mobile apps, up-to-date content is an absolute must and developers are without doubt required to leverage the power of the new always-on paradigm. If you're a mobile developer and would like to learn how to create and consume web services then this workshop is for you.

In this workshop, Paul will show you how to design and implement data-driven iOS apps using Amazon Web Services. Firstly, you will learn how to plan, build and deploy APIs using services such as EC2, RDS and S3, as well as understanding other Amazon services that can make your data available worldwide without performance penalties.

We will then use the APIs provided to build a simple iOS app. Some of the iOS topics we will cover include:

- Using storyboards to build user interfaces fast
- Data persistence using Core Data
- Threading for API consumption and interface updates
- Storing user data securely
- Testing in various network conditions
- Debugging and optimisation

Workshop ref: SA11

APPLIED BDD WITH CUCUMBER, CUCUMBER-JVM AND SPECFLOW

Seb Rose

It's all very well reading books, but nothing beats actually getting practical experience. In this workshop, working in your choice of Java, C# or Ruby, we will drive out the implementation of a simple utility by specifying its behaviour in Cucumber. The tyrannical Product Owner will regularly change his mind, so we'll need to keep our code well factored and easy to modify.

This session is designed for developers and testers. By the end of the day, you will be comfortable working with Cucumber in your chosen development environment. You'll have seen, first hand, how you can use Cucumber to drive out valuable features for your customers and how that can help keep your stakeholders engaged in the software development process. It will also be clear how BDD interacts with TDD, and how you will utilise both approaches to deliver software that does the right thing and does the thing right.

You should bring a laptop with your chosen development environment installed. And please try to pre-install your chosen Cucumber variant before you come (instructions available). But don't worry, we can install on the day, if necessary.

By the end of the session, you will have a good understanding of how Cucumber/Cucumber-JVM/SpecFlow work, and their place in the BDD toolkit.

If this subject interests you, but you would prefer a gentler introduction, then don't miss Seb's earlier workshop on Tuesday: "BDD by example".

Workshop ref: SA12

g my understanding from 20°
ormation packed. No sleepers!

2014 speakers

The Software Architect 2014 speakers are acknowledged experts in their field.

Recognised internationally, the 2014 speaker faculty comprises professional consultants, trainers, industry veterans, thought-leaders and published authors. Find out more about their experience and expertise here.

For comprehensive speaker biographies, please visit:
software-architect.co.uk/speakers

Paul Ardeleanu

Paul is an experienced iOS and web developer, trainer and mentor, specialising in complete software solutions, technical leadership and amazing customer experiences. He started programming back when Fortran was cool and is now using programming languages that have not yet been invented.

Austin Bingham

Austin is a founding director of Sixty North, a software consulting, training and application development company. He has previously worked at National Instruments, developing LabVIEW; at Applied Research Labs (University of Texas, Austin), developing sonar systems for the US Navy; and at a number of telecommunications companies.

Richard Blewitt

Richard has been working in the software industry for more than 20 years, starting with mainframes through the early years of client/server to today's service-oriented world. He has spent his professional life working on large distributed systems, including being the middle-tier architect on the UK national police systems.

Simon Brown

Simon works as an independent consultant, helping teams to build better software. His client list spans more than 20 countries, including organisations ranging from small technology start-ups through to global household names. He is an award-winning speaker and the author of "*Software Architecture for Developers*".

Andy Clymer

Andy is a co-founder of Rock Solid Knowledge. Prior to that, he cut his teeth working in various start-ups, programming on a host of platforms, finally working for a start-up bought by Cisco in 1997. He now spends his time working on RSK's Kiosk-based solutions on Windows Embedded with .NET.

Ed Courtenay

Ed is an experienced software developer and technical evangelist who has been programming professionally for more than 25 years. He currently works for a major manufacturer and retailer in the UK, leading the development team responsible for their ecommerce web site.

Howard Deiner

Howard is a software consultant and educator who specialises in agile process and practices. He has a varied background, spanning well over 30 years, with extensive experience in commercial software, aerospace and financial services. He has played many roles, such as developer, analyst, team lead, architect and project manager.

Dino Esposito

Dino is a trainer and software consultant based in Rome. A member of the IDesign team, he specialises in Microsoft .NET technologies, and spends his time teaching and consulting across Europe and the USA. He has hands-on experience in architecting and building distributed systems for banking and insurance companies.

János Feher

Since 1996, **János** has been involved in a wide variety of projects, including technical support for NATO operations, development of a high-performance computing grid, national TV and radio websites, and web applications for university and adult learning. János is currently the Head of Development for Intern Avenue.

Gil Fink

Gil is a web-development expert and ASP.Net/IIS Microsoft MVP. He works as a senior consultant and architect, and is currently consulting for various companies, where he helps develop web and RIA-based solutions. He conducts lectures and workshops for individuals and enterprises who want to specialise in infrastructure and web development.

Ido Flatow

Ido is a senior architect and trainer at SELA Group, a Microsoft ASP.NET/IIS MVP, and an expert on Microsoft Azure and web technologies such as WCF, ASP.NET and IIS. He is co-author of Microsoft's official courses for WCF 4 (10263A) and the manager of the Israeli Web Developers User Group.

Neal Ford

Neal is director, software architect and meme wrangler at ThoughtWorks, a global IT consultancy focusing on end-to-end software development and delivery. He is also the designer and developer of applications, magazine articles and video/DVD presentations. And author/editor of eight books, including the most recent "*Presentation Patterns*".

Shay Friedman

Shay is a Visual C#/ IronRuby MVP and the author of "*IronRuby Unleashed*". With more than 10 years of experience in the software industry, Friedman now works for CodeValue, a company he co-founded, where he creates products for developers, consults and conducts courses about web development and dynamic languages.

Nuni Filipe Godinho

Nuno is Director of Cloud Services, Europe at Aditi Technologies, and has more than 16 years' experience in IT and more than eight years in cloud computing. He regularly helps customers create a cloud strategy but also designs, plans, manages, develops and maintains products and solutions in the cloud.

Sasha Goldshtein

Sasha is the CTO of SELA Group, a Microsoft C# MVP and Azure MRS, and an international consultant and trainer. He is the author of "*Introducing Windows 7 for Developers*" (Microsoft Press, 2009) and "*Pro .NET Performance*" (Apress, 2012), a prolific blogger, and author of numerous training courses.

Allen Holub

Allen is an internationally recognised consultant, trainer, speaker and author. He specialises in lean/ agile processes and culture, agile-focused architecture and cloud-based web-application development. Allen has worn every hat from grunt programmer to CTO, and has written a dozen books and hundreds of magazine articles for various technical publications.

Sander Hoogendoorn

In his role of principal technology officer at Capgemini, **Sander** is concerned with the innovation of software development. He is also responsible for Capgemini's agile software development platform, and is recognised as an agile thought leader, as well as being a certified global software engineer (level 4) at Capgemini.

Michael Kennedy

As a full-time instructor for DevelopMentor, **Michael** specialises in core .NET, web, agile development, and test-driven development (TDD) technologies. He has been building commercial applications with .NET since its initial public beta in 2001. Prior to working with .NET, he spent years working with C++ on Windows and SGI platforms.

Ruth Malan

Having worked in the software architecture field since the mid-90s, **Ruth** has arguably played a pioneering role, helping to define architectures and the process by which they are created and evolved, as well as helping to shape the role of the software, systems and enterprise architect.

Jules May

Jules is a software architect with a particular interest in languages (both for programming and for discourse), and is presently active in web and mobile convergence. He has been writing, teaching and speaking for 25 years now, and conducts frequent lectures and workshops to a range of audiences. He is the originator of "Problem Space Analysis".

Brian A. Randell

Brian is a senior consultant with MCW Technologies, LLC. For more than 20 years, he has been building software solutions and educating his fellow developers. He spends his time teaching Microsoft technologies to developers, working with new and emerging technologies, and consulting worldwide.

Seb Rose

Seb wrote his first commercial software in the early 80s on an Apple II. He went on to graduate from the University of Edinburgh with a First-Class Joint Honours in Computer Science and Electronics. Over the past six years, he has focused on helping teams adopt and refine their agile practices.

Nathaniel Schutta

Nathaniel is a software architect focused on mobile and making usable applications. A proponent of polyglot programming, Nate has written two books on Ajax and speaks regularly at various worldwide conferences, No Fluff Just Stuff symposia, universities, and Java user groups.

Gary Short

Gary is a freelance data science practitioner and trainer, based in Dundee, UK. He has a deep understanding of the full Hadoop and HDInsight environment, as well as an interest in Social Network Analysis, (UCINet and Pajek) and computational linguistics (NLTK).

Robert Smallshire

Robert is a founding director of Sixty North, a software product and consulting business in Norway. He has worked in senior architecture and technical management roles for several software companies, providing tools in the energy sector for dealing with the masses of information flowing from today's digital oil fields.

Ralf Westphal

Ralf is a freelance consultant, project coach and trainer on software architectural topics and software team organisation. He is the co-founder of the German "Clean Code Developer" initiative, which aims to increase software quality, and currently counts more than 3,700 members.

Eoin Woods

Eoin is a lead architect in the Operations Technology group at UBS. Prior to UBS, he has worked in the software engineering field for more than 20 years, for a number of companies, including Bull, Sybase, InterTrust and BGI. His main technical interests are software architecture, distributed systems, computer security and data management.

Prices and sponsorship

To register, please visit software-architect.co.uk/book
For any registration enquiries, please contact the bookings team:
T: +44 (0)20 7407 9964 E: sarchitect2@bsi.co.uk
All prices include refreshments, buffet lunch and session notes but exclude travel and accommodation. If you require accommodation, you can secure discounted rates at recommended hotels via the Software Architect website: **software-architect.co.uk/about**

BOOK NOW
BOOK YOUR PLACE BY 1 AUGUST AND SAVE UP TO £200

PRICING OPTIONS

	Bookings made by Friday 1 August Save up to £250	Bookings made by Friday 19 September Save up to £100	Bookings made after Friday 19 September
Universal Pass [All 4 days]	£1,395 +VAT	£1,495 +VAT	£1,595 +VAT
Three-day Pass [Main Conference plus one Workshop day]	£1,095 +VAT	£1,195 +VAT	£1,295 +VAT
Main Conference [Wednesday & Thursday]	£795 +VAT	£895 +VAT	£995 +VAT
One Workshop Day [Tuesday or Friday]	£445 +VAT	£495 +VAT	£545 +VAT

Please note that the online submission of a completed registration form constitutes a firm booking, subject to the following terms and conditions. Any cancellations received after Friday, 7 July 2014, will incur a 30% administration fee. Cancellations must be made in writing at least 60 days before the conference, or the full fee will be charged. We are happy to accept substitutions if they are submitted in writing before the conference begins. The organisers reserve the right to make changes to the programme and speakers without notice, if this is unavoidable. If delegates are unable to attend for any reason that is beyond the control of the organisers, such as transport problems, personal illness, bereavement, inclement weather, terrorism or act of God, it will not be possible to make any refunds of conference or workshop fees.

SPONSORSHIP, EXHIBITION & MEDIA PARTNERSHIP OPPORTUNITIES

Software Architect offers you the opportunity to position your company as one of the leading players in the European software development market. With access to a senior-level audience of key influencers and decision-makers, from a wide range of industries, you have the perfect platform to generate awareness, showcase new products and services, and demonstrate your experience and expertise.

To discuss sponsorship & exhibition opportunities please contact:
Chris Handsley
+44 (0)20 7830 3634
chris.handsley@publicis-blueprint.co.uk