THE MICROSOFT JOURNAL FOR DEVELOPERS

JANUARY 2018 VOL 33 NO 1

msdn magazine

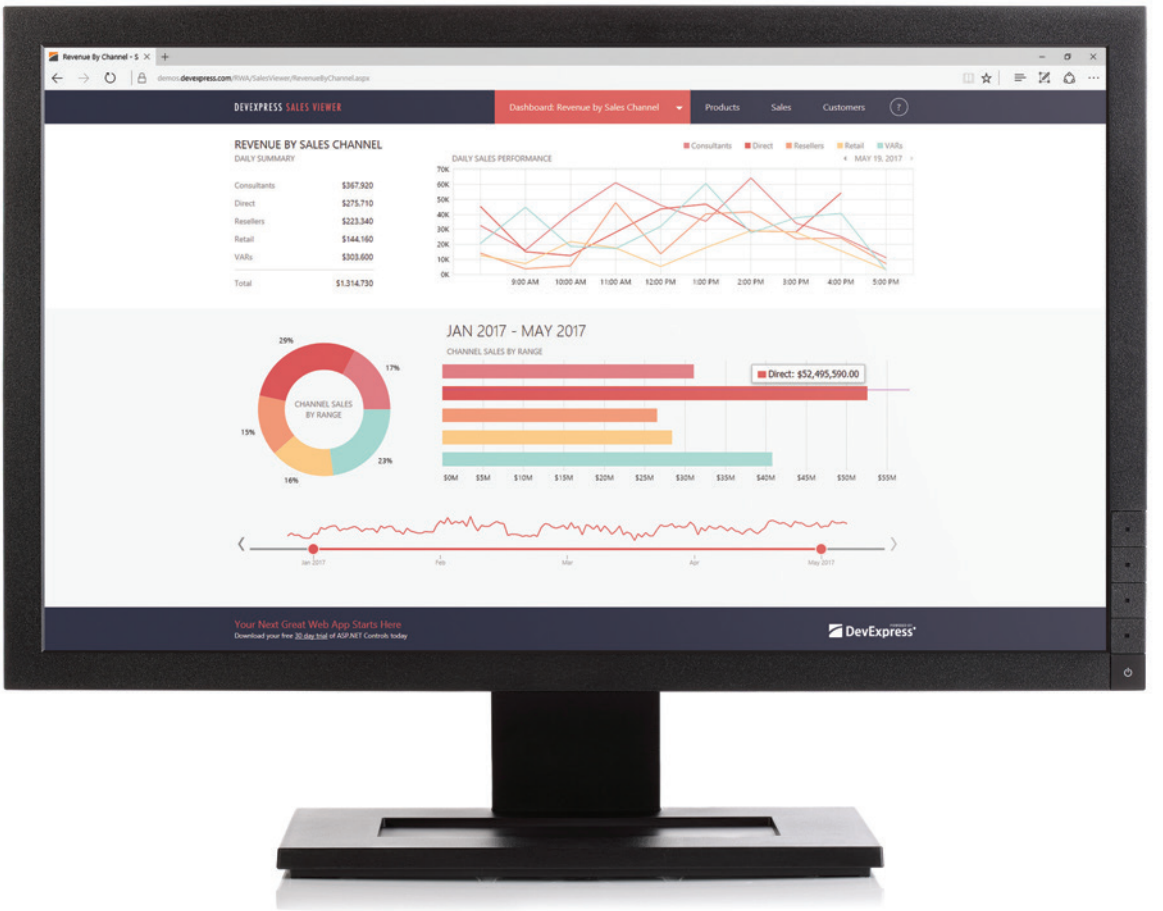
C#

Working with Span in C# 7.2.....22

DevExpress®

Your Next Great Web App Starts Here

From apps that replicate the look and feel of Microsoft Office® 365, to high-impact decision support systems for your enterprise, DevExpress Web Controls for ASP.NET will help you build your best, without limits or compromise.



Download your free 30-day trial
and experience the DevExpress difference today.

devexpress.com/try

THE MICROSOFT JOURNAL FOR DEVELOPERS

JANUARY 2018 VOL 33 NO 1

msdn magazine

C#

Working with Span in C# 7.2.....22

COLUMNS

Microsoft

INFRAGISTICS

Infragistics Ultimate 17.2

Productivity Tools & Fast Performing UI Controls for Quickly Building Web, Desktop, & Mobile Apps

Includes 100+ beautifully styled, high-performance grids, charts & other UI controls, plus visual configuration tooling, rapid prototyping, and usability testing.

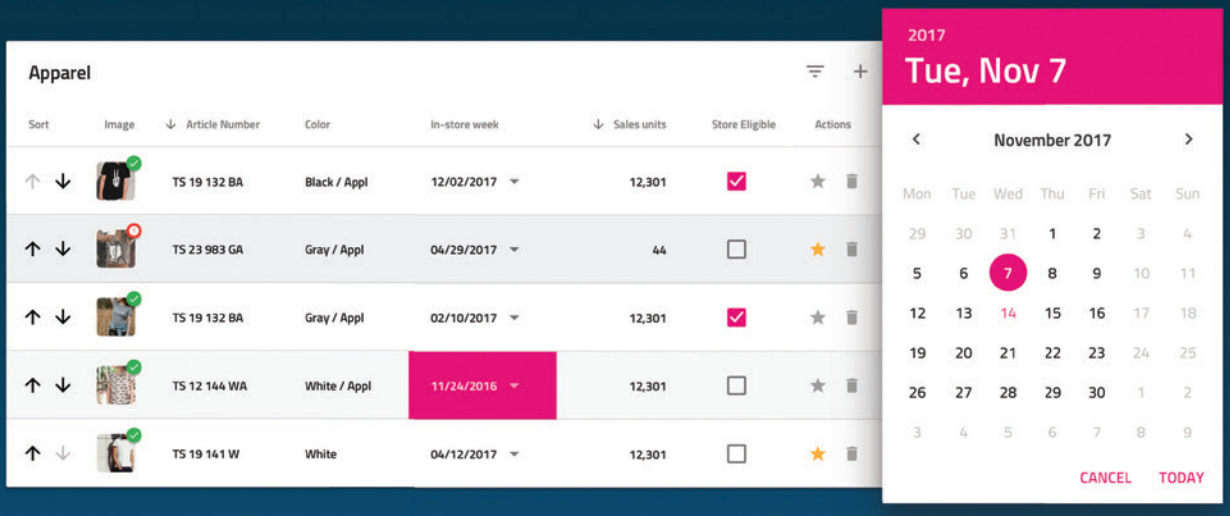
Angular | JavaScript / HTML5 | ASP.NET | Windows Forms | WPF | Xamarin

Download a free trial at
Infragistics.com/Ultimate

Featuring

Ignite UI

A complete UI component library for building high-performance, data rich web applications



- Create beautiful, touch-first, responsive desktop & mobile wep apps with over 100 JavaScript / HTML5, MVC & **Angular components**.
- Our easy to use Angular components have no 3rd party dependencies, a tiny footprint, and easy-to-use API.
- The Ignite UI **Angular Data Grid** enables you to quickly bind data with little coding - including features like sorting, filtering, paging, movable columns, templates and more!
- Speed up development time with responsive layout, powerful data binding, cross-browser compatibility, WYSIWYG page design, & built-in-themes.

Download a free trial of Ignite UI at: **Infragistics.com/Ignite-ui**

To speak with our sales team or request a product demo call: 1.800.321.8588

msdn magazine

JANUARY 2018 VOLUME 33 NUMBER 1

ID STATEMENT MSDN Magazine (ISSN 1528-4859) is published 13 times a year, monthly with a special issue in November by 1105 Media, Inc., 9201 Oakdale Avenue, Ste. 101, Chatsworth, CA 91311. Periodicals postage paid at Chatsworth, CA 91311-9998, and at additional mailing offices. Annual subscription rates payable in US funds are: U.S. $35.00, International $60.00. Annual digital subscription rates payable in U.S. funds are: U.S. $25.00, International $25.00. Single copies/back issues: U.S. $10, all others $12. Send orders with payment to: MSDN Magazine, P.O. Box 3167, Carol Stream, IL 60132, email MSDNmag@1105service.com or call (847) 763-9560. POSTMASTER: Send address changes to MSDN Magazine, P.O. Box 2166, Skokie, IL 60076. Canada Publications Mail Agreement No: 40612608. Return Undeliverable Canadian Addresses to Circulation Dept. or XPO Returns: P.O. Box 201, Richmond Hill, ON L4B 4R5, Canada.

CORPORATE ADDRESS 1105 Media, 9201 Oakdale Ave. Ste 101, Chatsworth, CA 91311 1105media.com

MEDIA KITS Direct your Media Kit requests to Chief Revenue Officer Dan LaBianca, 972-687-6702 (phone), 972-687-6799 (fax), dlabianca@1105media.com

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

LIST RENTAL This publication's subscriber list, as well as other lists from 1105 Media, Inc., is available for rental. For more information, please contact our list manager, Jane Long, Merit Direct. Phone: 913-685-1301;
E-mail: jlong@meritdirect.com;
Web: meritdirect.com/1105

Reaching the Staff

Staff may be reached via e-mail, telephone, fax, or mail.

E-mail: To e-mail any member of the staff, please use the following form: FirstinitialLastname@1105media.com

Irvine Office (weekdays, 9:00 a.m. – 5:00 p.m. PT)
Telephone 949-265-1520; Fax 949-265-1528
4 Venture, Suite 150, Irvine, CA 92618

Corporate Office (weekdays, 8:30 a.m. – 5:30 p.m. PT)
Telephone 818-814-5200; Fax 818-734-1522
9201 Oakdale Avenue, Suite 101, Chatsworth, CA 91311

The opinions expressed within the articles and other contentsherein do not necessarily express those of the publisher.

Microsoft

BPA WORLDWIDE

1906 100 2006 AMERICAN BUSINESS MEDIA

WPA Western Publications Association

Editor's Note

MICHAEL DESMOND

Going to Graph

It's been a challenge since forever: How do you unlock the data created in and captured by productivity applications such as Word, Excel, Outlook, SharePoint and OneNote? And just as important, how do you link up and imbue that data with context, so it can be intelligently interpreted, leveraged and consumed?

It's a question that Microsoft has spent years working to answer, in the form of Microsoft Graph. Graph is a family of rich, consistent REST APIs that provides vital connective tissue between applications, Azure cloud infrastructure, and IT resources and data stores. In our Connect(); special issue last year, Microsoft Principal Program Manager Yina Arenas explored the Graph APIs in her article, "Microsoft Graph: Gateway to Data and Intelligence" (msdn.com/magazine/mt790189). There, she described Graph as "the unified gateway for developers to access all the data, intelligence and APIs housed in Microsoft's intelligent cloud, including Exchange, SharePoint, Azure Active Directory, OneDrive, Outlook, OneNote, Planner, Excel and more."

She went on to describe how the intelligent Graph engine uses machine learning to provide calculated insights and rich relationships. But the most important thing is that this capability is accessed via a single REST API endpoint, vastly simplifying developer interaction across all of Microsoft's APIs by bringing them together, as Arenas wrote, "in a single URI namespace with a single authentication story."

A year later, Microsoft continues to refine, improve and extend its Graph APIs. In this issue of *MSDN Magazine*, Mike Ammerlaan writes "Build the API to Your Organization with Microsoft Graph and Azure Functions," in which he describes how the new Azure Functions Binding Extensions can be used to automate common tasks and processes. In the article, Ammerlaan walks through using Binding Extensions to pull together disparate sets of data in one piece of code using a single authentication token. He shows how the extensions can be used to access and prepare files stored in Microsoft OneDrive, and then perform voice recognition via Microsoft Cognitive Services.

As director of product marketing on the Microsoft Office Ecosystem team, Ammerlaan says the Graph APIs are the result of a years-long effort that demanded a strategic rethink across teams at Microsoft.

"Building Microsoft Graph required coordinating API designs and consistency across a dozen teams (and growing) at Microsoft, allowing for quick evolution while building a more universal and consistent API."

Ammerlaan says the Graph APIs are the result of a years-long effort that demanded a strategic rethink across teams at Microsoft.

For organizations looking to migrate to Graph, Ammerlaan suggests a measured approach. He notes that many existing product-specific APIs like SharePoint REST continue to be actively updated. So while Graph APIs are conceptually similar to existing product APIs, the decision to shift to Graph depends on a lot of factors.

"For new projects, I think developers should strongly consider evaluating Microsoft Graph and seeing what facets of it may apply to new projects," says Ammerlaan. "The breadth of tools, SDKs, and documentation make Microsoft Graph the best way to access Microsoft data and insights. I'm convinced every enterprise application could benefit from Microsoft Graph to be at its most effective."

Over time, Ammerlaan says, the benefits of adopting Microsoft Graph compound with investments into the APIs. As he writes in the article: "The more APIs you can bring together and connect, the more useful by far the net set of products you build could be—greater than the sum of its parts."

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

DevExpress®

DevExpress Spreadsheet for WPF & WinForms

with Chart, Pivot Table and Cell Editor support

Your users already know Microsoft® Excel® and with the new capabilities we've introduced in our Spreadsheet Control, you can create solutions that utilize a "spreadsheet-first" UX for a broad range of use-case scenarios. Imagine sales entry forms that are an actual invoice or an inventory management document with built-in ordering and data entry options.

Give our Spreadsheet a try today and see how easy it is to build Windows® apps that amaze.

Free 30-day trial

devexpress.com/spreadsheet

#UseTheBest

Crisis of Confidence

I'm often asked by engineers how they can overcome their "lack of confidence" in taking their next big professional leap. In this column, I'll answer this in two parts: First, by examining confidence itself, and second, by discussing what to do if, after you take the leap, things start to fall apart.

Part I. The Reframe

Feeling lack of confidence is very common, and it can stunt professional growth. Here's how you can overcome it.

Doubt it. Sometimes, when engineers say "lack of confidence," what they really mean is that they feel *stuck*. They have believed certain thoughts, acted certain ways, and associated with certain people for so long that their identity has become the sum of their opinions, actions, and friendships.

To break out of the pattern, think opposite thoughts, act in opposite ways and associate with people you wouldn't normally associate with. The unfamiliarity can jolt the system so much that you temporarily shed your conditioning, and in that space, a new self-identity can emerge.

Catch it. The fastest way to grow confidence is to be around positive, optimistic, self-assured people. You may be running low on confidence because of coworkers, friends, TV personalities, or online acquaintances who are self-flagellating or unkind to themselves, because that's *their* conditioning.

Until you build up your own emotional reserves, such interactions can drain you. Fortunately, the cure is simple: you seek out the company of those with a higher self-image. If you don't know any, read the books and watch the videos of those who do. Carefully self-monitor your exposure to draining people; it's not your job to fix or teach them, directly or indirectly.

Play it. You can ask yourself, "If I were supremely confident in this moment, what tiny step would I take right now?" Then, go do it! When you do this in small ways here and there, you begin to realize something: Confidence is the natural outcome of hundreds of small acts of courage.

Part II. The Post-Confidence Funk

Once you take the leap, you may make sudden, dramatic strides in your productivity and your outlook toward people. You may taste success that had seemed a struggle, but now seems easy.

Yet, not long after, you may find yourself in a funk—it's as if the growth you experienced never even happened. In fact, your original problems may suddenly get a lot worse. Was that growth spurt a fluke? Why is it all falling apart now, just when you were finally getting your bearings?

As an example, an engineer who's wasting all his time at work on the Internet may finally admit that he has a problem, and seek help from a friend. That's excellent growth! Yet, he may notice that he now starts binging on the Internet like he's never binged before. His old problem is back, bigger than ever. The engineer then connects his latest binge to the act of him seeking help. He panics, and *stops* seeking help! He reasons that at least his old problem wasn't *this* bad before.

You may see this in your own careers. You've just uncovered a big insight, and taken a positive next step. But right at this crucial moment, you may feel suddenly overwhelmed, even out-of-control. So, you stop, and go back to the old ways.

Behind the Scenes

This process is more common among those who are sensitive and introverted, so it's something software engineers often encounter. Basically, anytime you take on a new identity or set of behaviors, you set up an internal battle between who you're trying to be—Version 2 (V2)—and who you no longer want to be—Version 1 (V1).

The process of unlearning your V1 habits creates a struggle with older parts of yourself that are used to seeing *only* V1. These older parts don't yet understand what's happening. They may even feel that they're being attacked. These older parts have your best interests at heart, but they lack perspective. So, they try to bring back the only friend they've ever known—V1—by any means necessary.

They'll bring back older desires and compulsions with greater force, with greater urgency, because those were hallmarks of V1. They don't yet know that V2 is just a healthier upgrade to V1. They haven't gotten to know V2 yet.

An Answer

The solution is deceptively simple. No matter how many times you fall down in your quest to become V2, you just get back up, and keep trying again. Every time you do this, a little more of you—your will, your desires, your personality, different parts of you—shifts loyalties from V1 to V2.

Over a period of years, you may reach a point where a little more of you lives in V2 than in V1. And when that happens, you're no longer running away from V1; you're striding toward V2. ■

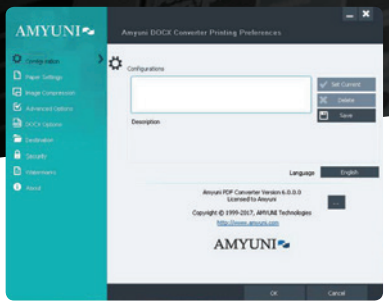
KRISHNAN RANGACHARI *helps engineering managers have more impact. Visit RadicalShifts.com for his newsletter and coaching.*

DOCXCONVERTER
For Windows

Free Demo at DOCXConverter.com

Amyuni DOCX Converter for Windows

Convert any document, including PDF documents, into DOCX format.
Enable editing of documents using Microsoft Word or other Office products.



A standalone desktop version, a server product for automated processing or an SDK for integration into third party applications.

Create

Create naturally editable DOCX documents with paragraph formatting and reflow of text

Convert

Convert images and graphics of multiple formats into DOCX shapes

OCR

Use OCR technology to convert non-editable text into real text

Extract

Extract headers and footers from source document and save them as DOCX headers and footers

Open

Open PDF documents with the integrated PDF viewer and quickly resave them to DOCX format

Configure

Configure the way the fonts are embedded into the DOCX file for optimal formatting

A virtual printer driver available for Windows 7 to Windows 10 and Windows Server 2008 to 2016

Powered by Amyuni Technologies:

Developers of the Amyuni PDF Converter and Amyuni PDF Creator products integrated into hundreds of applications and installed on millions of desktops and servers worldwide.

www.docxconverter.com

Creating Azure Functions to Interact with Cosmos DB

In my last column, I walked you through building a simple Universal Windows Platform (UWP) app—a game called CookieBinge—that was basically an excuse for me to explore using Entity Framework Core 2.0 (EF Core) in a device-bound Windows 10 app. The addition of providers for portable databases such as SQLite complete the picture to make this a possibility.

The app, in its current state, stores game scores locally on the device where the game is being played by using EF Core 2.0 to persist the data to a SQLite database. In that last article, I promised the next iteration of this game would allow users to share that data with other game players on the Internet. To achieve that goal, I'll be using two cool features of Azure: Azure Cosmos DB and Azure Functions.

A Bit About Azure Cosmos DB

Azure Cosmos DB is the next generation of what began life as Azure Document DB—a technology I've written about numerous times in this column. "An Overview of Microsoft Azure DocumentDB" (msdn.com/magazine/mt147238) in June 2015 was followed by two more articles where I used it as the back end of an Aurelia Web app by way of a Node.js server API.

DocumentDB evolved to become a globally distributed database with some extraordinary features. Besides the ease with which it can be distributed globally, its consistency models have been realigned so that there's more to choose from than just strong and eventual. Between those two extremes, users can now also choose bounded-staleness, session or consistent prefix. There are many more important features to support the data store, and the document database is now joined by a number of other data models—documents accessible via MongoDB APIs, graph databases, table (key/value pair) and column data storage. All of these models are under the Cosmos DB umbrella. In fact, if you had an Azure DocumentDB, it was automatically switched to an Azure Cosmos DB, allowing your existing document database to benefit from all of the new features Cosmos DB brings. You can read much more about Cosmos DB by starting at cosmosdb.com.

And a Bit About Azure Functions

I'm going to use Cosmos DB to store the CookieBinge scores. However, rather than write all of the code myself using the Document DB APIs, I'll take advantage of another relatively new feature of Azure: Azure Functions. Azure Functions is the Microsoft "serverless computing" offering. I've always been a skeptic about that phrase because the computing is still on a server ... just not my server. But having finally had a chance to work with Azure Functions, I now have a great respect for the concept. Azure Functions lets you focus on the actual logic you want to perform in the app, while it takes care of the cross-cutting concerns such as deployment, supporting APIs, and connections to other functionality such as data storage or sending e-mails. I didn't totally understand this until I did it myself, so my hope is that by following my path as I prepare this feature for the CookieBinge app, you'll also have your aha moment.

Code download available at msdn.com/magazine/0118magcode.

Preparation for Building the First Azure Function

There are three ways to build Azure Functions. One is with tooling in Visual Studio 2017. Another is directly in the Azure portal. You can also use Visual Studio Code in combination with the Azure command-line interface (CLI). I decided to start my learning by way of the portal because it walked me through all of the steps I needed. Another benefit is that without the help of the Visual Studio 2017 tooling, I was forced to think a little harder about all of the moving parts. I feel I got a much better understanding that way. Of course, there are lots of fabulous resources for doing this in Visual Studio 2017, as well, but the other thing I like about using the portal is that it's Web-based and, therefore, a cross-platform option. Keep in mind, while you can deploy your function code and assets from source control into Azure, anything you build directly in the portal will have to be downloaded to your machine (a simple task) and from there, pushed to your repository.

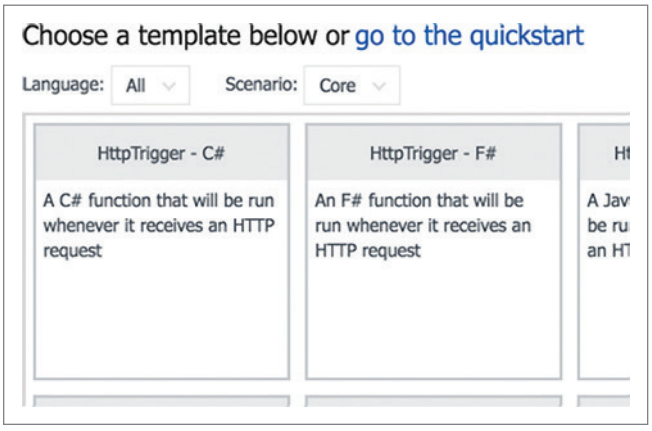


Figure 1 **A Bit of the Template List for Creating New, Custom Azure Functions**

run.csx Save ▶ Save and run </

```
1  using System.Net;
2
3  public static async Task<HttpResponseMessage> Run(HttpRequestMessage req,
4                                                    TraceWriter log)
5  {
6      log.Info("C# HTTP trigger function processed a request.");
7
8      // parse query parameter
9      string name = req.GetQueryNameValuePairs()
10         .FirstOrDefault(q => string.Compare(q.Key, "name", true) == 0)
11         .Value;
12
13     // Get request body
14     dynamic data = await req.Content.ReadAsAsync<object>();
15
16     // Set name to query string or body data
17     name = name ?? data?.name;
18
19     return name == null
20         ? req.CreateResponse(HttpStatusCode.BadRequest,
21           "Please pass a name on the query string or in the request body")
22         : req.CreateResponse(HttpStatusCode.OK, "Hello " + name);
23 }
24
```

Figure 2 **Default Function Logic for a New HTTPTrigger**

If you want to follow along and don't already have an Azure subscription, I'm happy to inform you that you can get a free subscription, and it's not just for a short trial. Some Azure products will be free for a year and there are a few dozen that will always be free. Go to azure.com/free to get set up. I'm using the account I get as part of my Visual Studio subscription, which has a monthly credit allowance for experimenting with Azure.

Before creating the functions, I need to define my goals. I want my app to be able to:

- Store user scores on the Web, persisting not only some user information and the date along with the score, but also the type of device the game was played on.
- Allow users to retrieve their highest scores across all of the devices on which they're playing.
- Allow a user to retrieve the highest scores across all users.

I'm not going to bog this lesson down with matters like creating and authenticating accounts, though of course you'd need that for the real world. My aim is to show you Azure Functions and, eventually, the interaction from the UWP app.

Creating the Azure Function in the Azure Portal

Azure Functions is a service that's grouped in a function app that allows you to define and share settings across its set of functions. So, I'll start by creating a new function app. In the Azure portal, click New and filter on "function app" to easily find that option. Click Function App in the results list and then Create, which prompts you to fill out some metadata, such as a name for your app. I named mine cookiebinge.azurewebsites.net. As this is just a simple demo, I'll accept the rest of the defaults on the Create page. For easy future access to your new function app, check the Pin to Dashboard option and then the Create button. It took only about 30 seconds for my new function app's deployment to be completed.

Now you can add some functions into the function app. My functions will be built to support the list of goals mentioned earlier. The Azure Functions service has a pre-defined (and quite rich) set of events it can respond to, including an HTTP request, a change in a Cosmos DB database, or an event in a blob or a queue. Because I want to call into these functions from the UWP app, I want functions that respond to requests coming over HTTP. The portal provides a slew of templates in a variety of languages: Bash, Batch, C#, F#, JavaScript, PHP, PowerShell, Python and TypeScript. I'll use C#.

To create the first function inside the function app, click on the plus sign next to the Functions header. You'll see buttons to create pre-defined functions, but if you scroll down below those buttons, you'll find a link to create a custom function. Choose that option and you'll see a scrollable grid filled with template options, as shown in **Figure 1**. HTTP Trigger – C# should be at the top of that list and that's what you should select.

Name the function and then click the Create button. I named mine StoreScores.

The portal will create a function with some default code so you can see how it's structured. The function is built into a file called run.csx (see **Figure 2**). You can have additional logic in supporting files, as well, but that's more advanced than needed for this first look.

The only method in the example is called Run, which is what Azure will call in response to an HTTP request to this function. It has one parameter to capture the request and another for relaying information to a log.

In the sample, you can see that the function is looking for incoming data that represents a name, and the function is flexible

Figure 3 **Defining the Functions Integration Points**

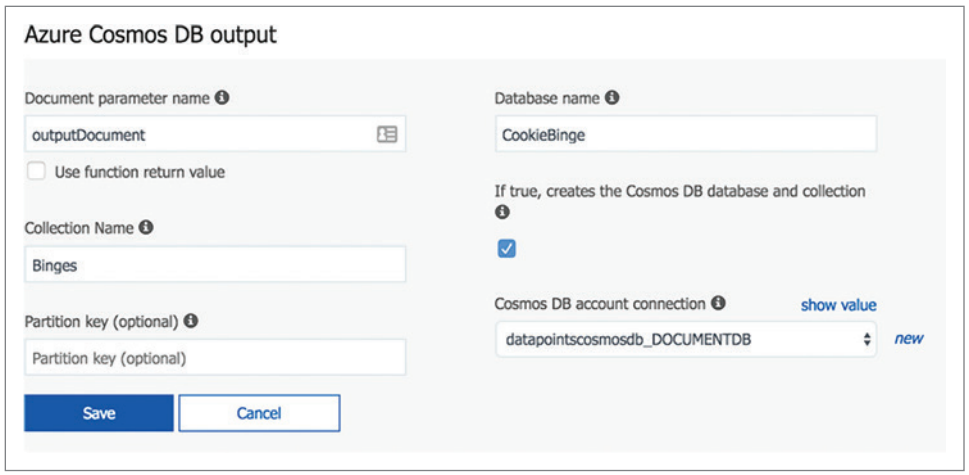


Figure 4 **Defining a Cosmos DB as Output for the Function**

enough to search for it in the query parameters and in the request body. If the name isn't found, the function will return an HttpResponseMessage with a friendly error message, otherwise it returns "Hello [name]" in the response.

Customizing the Function to Interact with Cosmos DB

The goal of the function is to store the incoming data into a Cosmos DB database. Here's where the magic begins. There's no need to create connections and commands and other code to do this task. Azure Functions has the ability to integrate easily with a number of other Azure products—and Cosmos DB is one of them.

> Azure Functions has the ability to integrate easily with other Azure products—and Cosmos DB is one of them.

In the Functions list, you should see your new function and three items below it. One of those items is Integrate. Select that and you'll see the form partially shown in **Figure 3**. Notice that it says the trigger is an HTTP request and that the output returns something through HTTP. Because I want to return a success or failure message, I do want to keep that HTTP output. But I also want to add an output that has a Cosmos DB collection as its destination.

To do this, click on New Output, which will display a list of icons. Scroll down to the one called Azure Cosmos DB, select it and then further down on the page you'll see a SELECT button. You know what to do. (Click that button!)

The screen to set up this integration gets pre-populated with defaults. The Document Parameter Name represents the parameter you'll use in run.csx. I'll leave that as the default name, outputDocument. Next are the names of the Cosmos DB database and the collection within that database, as well as the connection to the Cosmos DB account where the database lives. You'll also see a checkbox to automatically create the database for you. I already have a few Cosmos DB accounts created, so I'm going to use one of those, but I let my function create a new database named CookieBinge with a collection called Binges in that account. **Figure 4** shows how I've filled out this form before saving the output definition. Because I marked the checkbox to create the database and collection, those will be created for me, but not when I save this output. When the function first attempts to store data into the database and sees that it doesn't exist, the function will create the database on the fly.

Customizing Run.csx

Now it's time to redefine the function code. The new version of the function expects a JSON object passed in that aligns with this BingeRequest class, which I added into the run.csx file below the Run method:

```
public class BingeRequest{
public string userId {get;set;}
public string userName {get;set;}
public string deviceName {get;set;}
public DateTime dateTime {get;set;}
public int score{get;set;}
public bool worthit {get;set;}
}
```

But this is not the same structure as the data I want to store because I want to capture one more property—the date and time the data is logged into the database. I'll do that using a second class, BingeDocument, which inherits from BingeRequest, thereby inheriting all of its properties, as well as adding one more property named logged. The constructor takes a populated BingeRequest and after setting the value of logged, it transfers the BingeRequest values to its own properties:

Figure 5 **The New run.csx File Capturing the Binge and Storing It into the Output, Cosmos DB**

```
using System.Net;
public static async Task<HttpResponseMessage> Run(HttpRequestMessage req,
  TraceWriter log,    IAsyncCollector<object> outputDocument)
{
  BingeRequest bingeData =  await req.Content.ReadAsAsync<BingeRequest>();
  log.Verbose("Incoming userId:" + bingeData.userId);
  var doc=new BingeDocument(bingeData,log);
  log.Verbose("Outgoing userId:" + doc.userId);
  await outputDocument.AddAsync(doc);
  if (doc.userId !=" " ){
    return req.CreateResponse(HttpStatusCode.OK,$"{doc.userId} was created" );
  }
  else {
    return req.CreateResponse(HttpStatusCode.BadRequest,
      $"The request was incorrectly formatted." );
  }
}
public class BingeRequest{ . . . }
public class BingeDocument { . . . }
```

Figure 6 **The Log Window Displaying Compiler Information Including Errors**

```
public class BingeDocument:BingeRequest
  {
    public BingeDocument(BingeRequest binge){
    logged=System.DateTime.Now;
    userId=binge.userId;
    userName=binge.userName;
    deviceName=binge.deviceName;
    dateTime=binge.dateTime;
    score=binge.score;
    }
    public DateTime logged{get;set;}
  }
```

With these types in place, the Run method can take advantage of them. **Figure 5** shows the modified listing for run.csx, including placeholders for the BingeRequest and BingeDocument classes described earlier.

Let's parse the new Run method. Its signature takes a request and a TraceWriter just as the original signature did, but now it also has an asynchronous output parameter named outputDocument. The result of the output parameter is what will get pushed to the Cosmos DB output I defined. Notice that its name aligns with the output parameter name in the output configuration in **Figure 4**. The TraceWriter lets me output messages to the log window that's below the code window. I'll take these out eventually, but it's like the old days, without the IDEs that let you debug. Don't get me wrong, though. The code window is amazing at parsing the language you're working in, and when you save, any compiler errors, which are very detailed, also get output to the debug window. It also does things like inserting a closing brace when you type in an opening brace. There are an impressive number of editor features, in fact. For example, right-click on the editor window to see the long list of editor features of which you can take advantage.

The first line of code in Run asynchronously reads the request and converts its result into a BingeRequest object.

Next, I instantiate a new BingeDocument, passing in the object I just created from the request, which results in a fully populated BingeDocument object, along with the logged property populated.

> The code window is amazing at parsing the language you're working in, and when you save, any compiler errors, which are very detailed, also get output to the debug window.

I then use the TraceWriter to show some data from the request in the logs so, when debugging, I can tell if BingeDocument did indeed get the data from the request object.

Finally, I asynchronously add the BingeDocument I just created into the asynchronous outputDocument.

The result of that outputDocument object is what gets sent to Cosmos DB by the function. It gets stored into the DocumentDB as JSON—the function converts it again in the background for you. Because I wired everything up with the integration settings, I don't have to write any code to make that happen.

When all is said and done, I return a message via HttpResponse, relaying the function's success or failure.

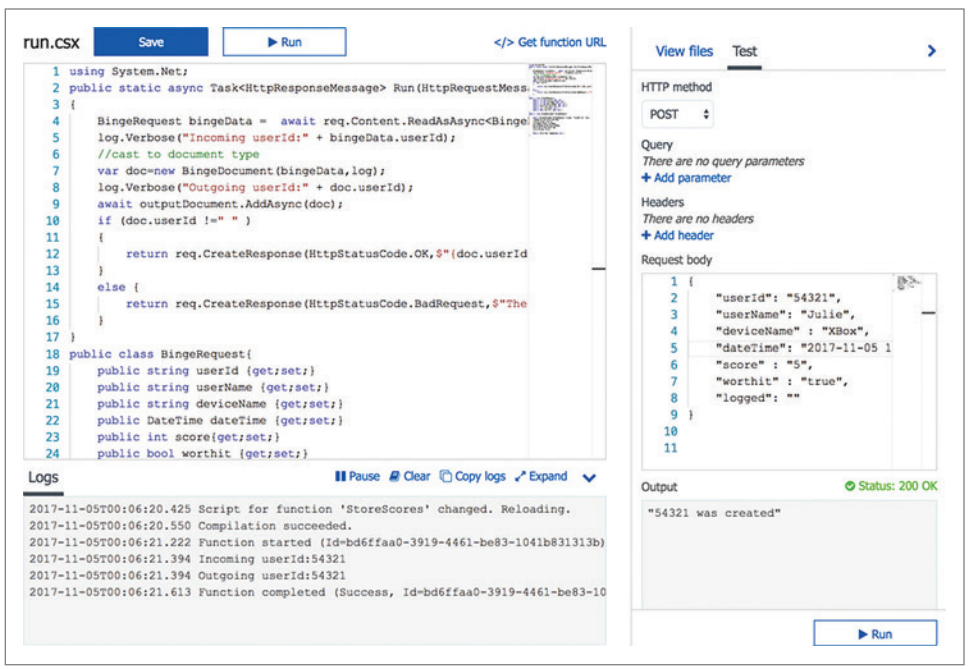


Figure 7 **The Test Pane After Running a Test on the Function**

Compiling and Testing the Function

Functions get compiled when you save them. I'm going to randomly delete something important in the code so you can see the compiler in action, with results being displayed in the log window below the code window. **Figure 6** shows the compiler output, highlighting the chaos I created by deleting an open brace from line 13. Even without a debugger, I've found that

File Format APIs

Open, Create, Convert, Print and Save files from your applications!

Try risk free - 30 day trial

Aspose.Total

Enable your applications to manipulate Word, Excel, PDF, PowerPoint, Outlook and more than 100 other file formats for all major platforms.

Aspose.Words

Create, edit, convert or print Word documents (DOC, DOCX, RTF etc.) in your .NET, Java and Android applications.

Aspose.Cells

Develop high performance .NET, Java and Android applications to Create, Edit or Convert Excel worksheets (XLS, XLSX, ODS etc).

Aspose.Pdf

Manipulate PDF file formats (PDF, PDF/A, XPS etc.) using our native APIs for .NET, Java and Android platforms.

Aspose.Slides

Create, edit or convert PowerPoint presentations (PPT, PPTX, ODP etc.) in your .NET, Java and Android applications.

Aspose.Email

Create, Edit or Convert Outlook Email file formats (MSG, PST, EML etc.) and popular network protocols.

Aspose.BarCode

Generate or recognize barcodes (Code128, PDF417, Postnet etc.) using our native APIs for .NET and Java.

Aspose.Imaging · Aspose.Tasks · Aspose.OCR · Aspose.Diagram · Aspose.Note · Aspose.HTML

ASPOSE File Format APIs

Download a Free Trial at
https://downloads.aspose.com

Americas: +1 903 306 1676 EMEA: +44 141 628 8900 Oceania: +61 2 8006 6987
sales@asposeptyltd.com

seeing these errors helps me work through code I've had to write without the aid of IntelliSense or other coding tools in my IDEs. At the same time, I learned how much I depend on those tools!

When the code is fixed up and the compiler is happy, the log will display the "Reloading" message followed by "Compilation succeeded."

Now it's time to test the function and you can do that right in the same window where you're coding the function. To the right of the code editor are two tabbed panes. One displays the list of files related to the function. By default, there are only two, the run.csx file I'm currently looking at and a function.json file that contains all of the settings defined in the UI. The other tab is for running tests. This built-in test UI is like a mini-Postman or Fiddler application for creating HTTP requests with a lot less effort because it's already aware of the function to be tested. All you need to do is insert a JSON object to represent the incoming request. The test UI defaults to sending an HTTP Post, so you don't even need to change that for this test. Enter the following JSON into the Request Body textbox. The schema is important, but you can use whatever values you want:

```
{
  "userId": "54321",
  "userName": "Julie",
  "deviceName" : "XBox",
  "dateTime": "2017-10-25 15:26:00",
  "score" : "5",
  "worthit" : "true",
  "logged": ""
}
```

Next, click the Run button on the Test pane. The test will call the function, passing in the request body, and then display any HTTP results in the Output window. In **Figure 7**, you can see the output "54321 was created," as well as the log output from the function in the Logs window.

Viewing the New Data in the Cosmos DB Database

What you can't see here is that as a result of this first successful test, the CookieBinge Cosmos DB database was created, and in it, the Binge collection where this document was stored. Let's take a look at that before wrapping up this installment of my multi-part column.

You can do this by first opening the Cosmos DB account in the portal where you created this database. Mine is called datapointscosmosdb, so I'll go to All Resources and type datapoints in the filter to find it. Once I open the account, I can see all of the collections and databases there, although the only one I have is my Binges collection in the CookieBinge database, as shown in **Figure 8**. That's what just got created by the function.

> This built-in test UI is like a mini-Postman or Fiddler application.

Click on Binges to open up the Data Explorer for that collection. I've run the test twice, so you can see in **Figure 9** that two documents were stored in the collection. The first seven properties in the document are the properties I defined. The rest are metadata that Cosmos DB and the relevant APIs use for tasks like indexing, searching, partitioning and more.

Looking Ahead

If you look back at **Figure 7**, you'll see there's a Get Function URL link above the code window. That URL is what I'll use in the CookieBinge app to send data to the cloud.

Now that you've seen how to create the function and hook it up with Cosmos DB, my next column will show you how to build two more functions to retrieve different views of the data. The final installment will show how to call the functions from the CookieBinge app and display their results. ■

Collections

ID	DATABASE	THROUGHPUT (RU/S)
Binges	CookieBinge	400

Figure 8 **The Binges Collection Listed in the datapointscosmosdb Account**

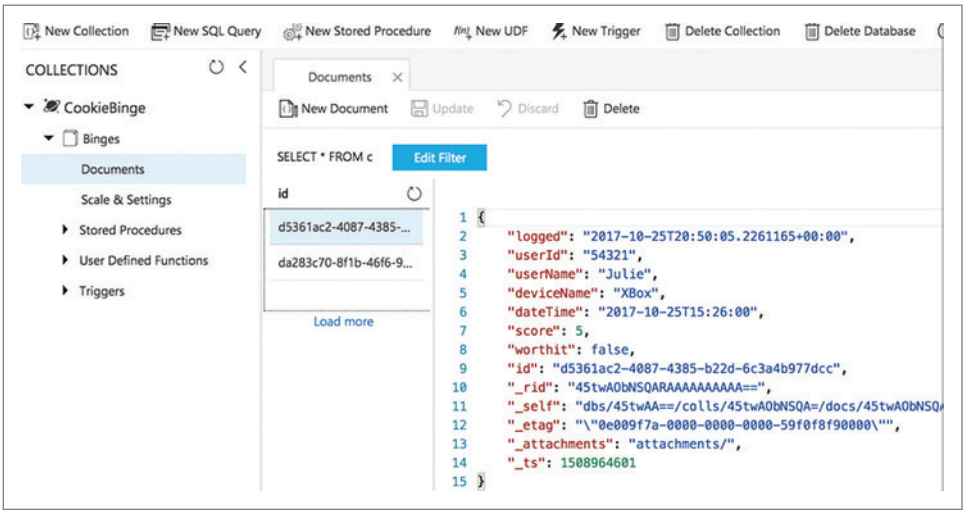


Figure 9 **Looking at the Stored Documents in Cosmos DB in the Portal**

Julie Lerman *is a Microsoft Regional Director, Microsoft MVP, software team coach and consultant who lives in the hills of Vermont. You can find her presenting on data access and other topics at user groups and conferences around the world. She blogs at the datafarm.com/blog and is the author of "Programming Entity Framework," as well as a Code First and a DbContext edition, all from O'Reilly Media. Follow her on Twitter: @julielerman and see her Pluralsight courses at juliel.me/PS-Videos.*

Thanks to the following Microsoft technical expert for reviewing this article: *Jeff Hollan*

Data Quality Made Easy.
Your Data, Your Way.

Melissa provides the full spectrum of data quality to ensure you have data you can trust.

We profile, standardize, verify, match and enrich global **People Data** – name, address, email, phone, and more.

Our data quality solutions are available on-premises and in the Cloud – fast, easy to use, and powerful developer tools and plugins for the **Microsoft® Product Ecosystem.**

Microsoft® SQL Server® Integration Services

Microsoft Dynamics® CRM

Microsoft® .NET

Microsoft Azure

Microsoft® Excel

Start Your Free Trial

www.melissa.com/msft-pd

Melissa Data is now Melissa.
See What's New at www.Melissa.com

1-800-MELISSA

melissa™

Creating Models in Azure ML Workbench

In my last column, I introduced Azure Machine Learning Workbench (Azure ML Workbench), a new tool for professional data scientists and machine learning (ML) practitioners. This stands in stark contrast to Azure Machine Learning Studio (Azure ML Studio), which is a tool primarily geared toward beginners. However, that doesn't mean Azure ML Workbench is only for experienced data scientists. Intermediate and even entry-level data scientists can also benefit from the tools provided in Azure ML Workbench.

Loading the Iris Classifier Project Templates

As noted in my previous column, Azure ML Workbench provides numerous project templates (I used the Linear Regression template). This time I'll utilize the Classifying Iris project template to demonstrate even more features of Azure ML Workbench. If you haven't already installed Azure ML Workbench, please refer to the documentation at bit.ly/2j2NVdH.

The Iris data set is a multi-variate data set that consists of 50 samples from each of three species of Iris. Four features were measured from each sample: the length and the width of the sepals and petals. Based on the combination of these four features, the species of Iris can be determined. It is an oft-used sample data set in data science and ML.

Open Azure ML Workbench, select Projects and click on the plus sign. In the context menu that appears, choose New Project to create a new project. Name the project IrisClassifier. Look for Classifying Iris in the Project Templates, click on it and click on the Create button (see **Figure 1**).

Viewing the Code

Once the project loads into Azure ML Workbench, click on the folder icon on the left side to reveal all the files included. Click on the iris_sklearn.py file to view its contents in the editor. It should look similar to **Figure 2**.

In case you were wondering, the code is in Python, a language popular with data scientists and ML practitioners. Python enjoys a diversity of ML, scientific and plotting libraries that provide the language with a rich ecosystem of tools and utilities. One of these is scikit-learn, a popular ML library. Various segments of the scikit-learn, referred to as sklearn in the code, are imported into the project in lines seven through 14. While a full tutorial on the Python language falls outside the scope of this article, the syntax should be familiar to any C# developer. The focus here will be on building models with scikit-learn.

> Intermediate and even entry-level data scientists can also benefit from the tools provided in Azure Azure ML Workbench.

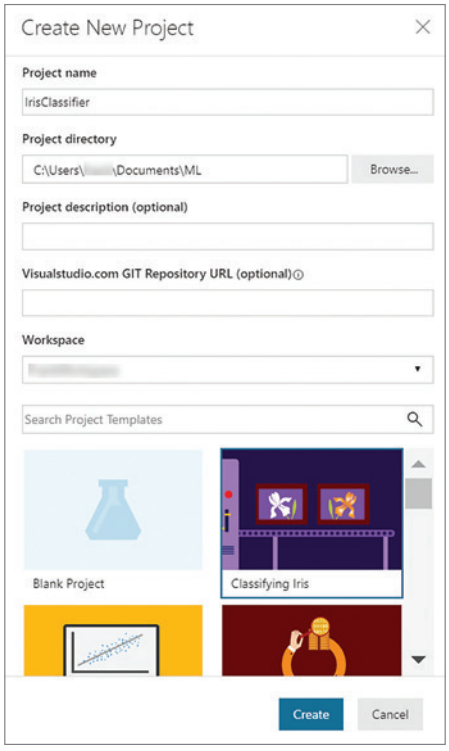


Figure 1 **Choosing the Classifying Iris Project Template**

Workflow of an Azure ML Workbench Project

The first step in any ML project is loading the data. The second step is often the more laborious and time-consuming: wrangling the data. This is where Azure ML Workbench really shines. Click on the iris.csv file to see what the raw data looks like. Note that this file lacks column names. Now, click on the iris.dprep file list on the left-hand side of the screen. Note the number of steps taken to clean the data. The steps add names to the columns and remove rows where the Species column is null. Click on the down arrow to the right of the Filter Species step. In the context menu that appears click Edit to display the Filter Column dialog shown in **Figure 3**. The rule is set up to remove any rows where the Species column is null. Note that there are additional options to add extra conditions. This dialog box will be an invaluable tool in your data science projects, as data rarely comes in a clean format that's consumable by ML algorithms.

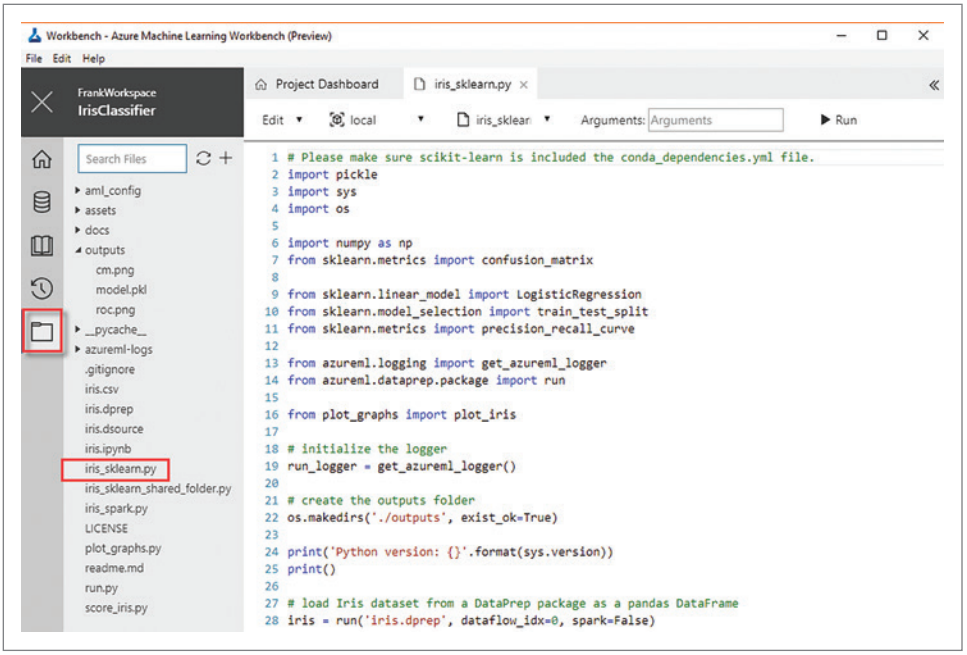


Figure 2 **The iris_sklearn.py File in the Azure ML Workbench Text Editor**

After referencing various libraries and initializing the local environment, the code loads the data by executing the iris.dprep file on line 28, which loads the data file and performs all the steps defined:

```
iris = run('iris.dprep', dataflow_idx=0, spark=False)
```

The first step in any ML project is loading the data.

The output is a pandas DataFrame with the cleaned data. A pandas DataFrame is a two-dimensional data structure similar to a table in a SQL database or in a spreadsheet. You can read more about DataFrames at bit.ly/2BIWI6K.

Now that the data has been cleaned and loaded, it's time to separate the data into features and labels. Features make up the various fields needed to make a prediction. In this case, given the widths and lengths for Sepals and Petals, the algorithm will predict to which species of Iris the plant belongs. In this case, the features are: Sepal Length, Sepal Width, Petal Length and Petal Width. The label, or predicted value, is the Species. Line 32 in the iris_sklearn.py file separates the DataFrame into two arrays: X for the features and Y for the label, as shown here (strictly speaking, X and Y are NumPy arrays, a data structure from the NumPy library):

```
X, Y = iris[['Sepal Length', 'Sepal Width', 'Petal Length', 'Petal Width']].values,
  iris['Species'].values
```

Once the data is separated into a label and features, it's time to separate the data into a training set and a test set. The following line of code randomly reserves 35 percent of the rows in the input data set and places it into Y_train and Y_test; the remaining 65 percent go into X_train and X_test:

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.35,
  random_state=0
```

In supervised learning, the correct values to the label are known. The algorithm is trained by passing the features and label to it. The algorithm then discovers the relationships and patterns between the features and the correct label. The following line creates an ML model with a logistic regression algorithm against the training data:

```
clf1 = LogisticRegression(C=1/reg).fit(X_train, Y_train)
```

Logistic regression is a statistical method analyzing data sets where there are one or more variables that determine an outcome (bit.ly/2zQ1hVe). In this case, the dimensions of the sepals and petals determine the Iris species.

Once trained, the algorithm is tested for accuracy by calling the score method on the model:

```
accuracy = clf1.score(X_test, Y_test)
print ("Accuracy is {}".format(accuracy))
```

The best way to understand this is to actually run the code. However, before that can be done, there's one more step. This project uses matplotlib, a popular plotting library for Python. To install it, select Open Command Prompt from the File menu. At the command line, type the following:

```
pip install matplotlib
```

Once installed, type the following to the command line:

```
python iris_sklearn.py
```

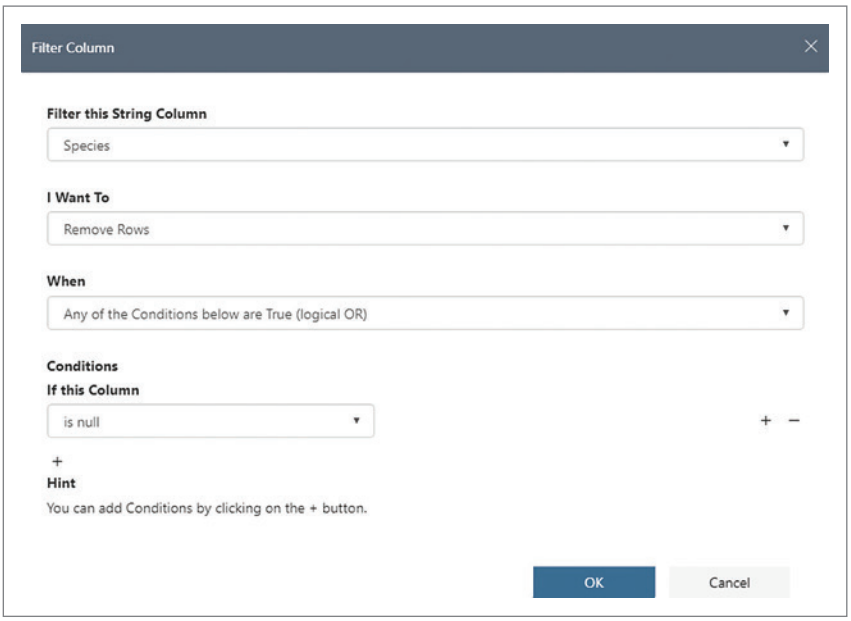


Figure 3 **The Filter Column Dialog Window**

In a few moments, the output should look like **Figure 4**.

As displayed in the command window, the accuracy of the model is 0.6792452830188679, meaning that it correctly guesses the species of Iris in the test data 67.92 percent of the time.

```
C:\Users\frank\Documents\ML\IrisClassifier>python iris_sklearn.py
Python version: 3.5.2 |Continuum Analytics, Inc.| (default, Jul  5 2016, 11:41:13) [MSC v.1900 64 bit (AMD64)]

Iris dataset shape: (150, 5)
Regularization rate is 0.01
LogisticRegression(C=100.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
Accuracy is 0.6792452830188679

======================================
Serialize and deserialize using the outputs folder.

Export the model to model.pkl
Import the model from model.pkl
New sample: [[3.0, 3.6, 1.3, 0.25]]
Predicted class is ['Iris-setosa']
Plotting confusion matrix...
Confusion matrix in text:
[[50  0  0]
 [ 1 37 12]
 [ 0  4 46]]
Confusion matrix plotted.
Plotting ROC curve....
ROC curve plotted.
Confusion matrix and ROC curve plotted. See them in Run History details page.
```

Figure 4 **Output of the iris_sklearn.py Program**

Executing the Code from Within Azure ML Workbench

While running the code in the command line is useful, Azure ML Workbench provides a way to make this simpler and capture information about the jobs that have run. Look for the Run button. To the immediate left of it, there are two dropdowns and a textbox. By default, it should look like **Figure 5**. Click Run.

Figure 5 **Running Files in Azure ML Workbench**

This executes the script locally and tracks the execution of the script via the Jobs tab in Azure ML Workbench. After the program runs, output and details about the run will appear. To see that, click on the iris_sklearn.py entry in the run list. Choose the first record in the data grid under Runs. Review the Run Properties section to see basic performance statistics of the run. Scroll down to see the Metrics and Visualization sections to see the output from the script as shown in **Figure 6**.

In my previous article, I explained how to explore the results of a job and view the job history. Please refer to that for more details (msdn.com/magazine/mt814414).

Persisting a Trained Model

While running the iris_sklearn.py script either through Azure ML Workbench or the command line, you'll likely notice that the process takes several seconds. On my Surface Book, it takes about nine seconds. While different hardware configurations will produce different results, the process is hardly instantaneous. Most of the processing time is devoted to training the model. Fortunately, there's little need to continually train a model. Lines 79 through 82 take the trained model and persist it to disk using the Pickle library (bit.ly/2im9w30):

```
print ("Export the model to model.pkl")
f = open('./outputs/model.pkl', 'wb')
pickle.dump(clf1, f)
f.close()
```

Lines 86 and 87 demonstrate how to restore the trained model from disk:

```
f2 = open('./outputs/model.pkl', 'rb')
clf2 = pickle.load(f2)
```

The next step is to create some sample data and use the model to predict the species, which is done on lines 89 and 98:

```
# Predict on a new sample
X_new = [[3.0, 3.6, 1.3, 0.25]]
print ('New sample: {}'.format(X_new))

# Add random features to match the training data
X_new_with_random_features = np.c_[X_new, random_state.randn(1, n)]

# Score on the new sample
pred = clf2.predict(X_new_with_random_features)
print('Predicted class is {}'.format(pred))
```

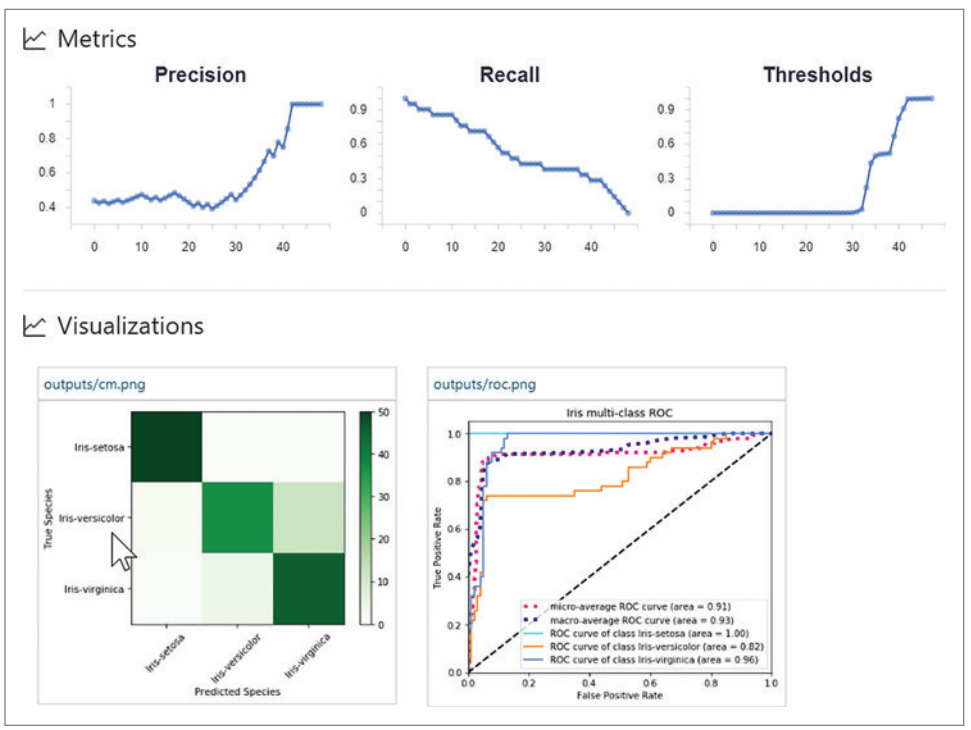
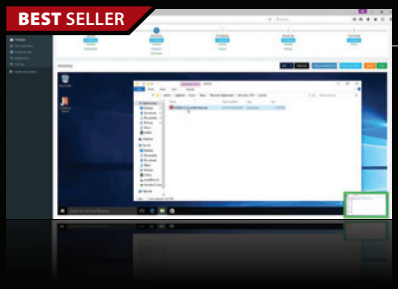


Figure 6 **Metrics and Visualizations Shown in Azure ML Workbench**

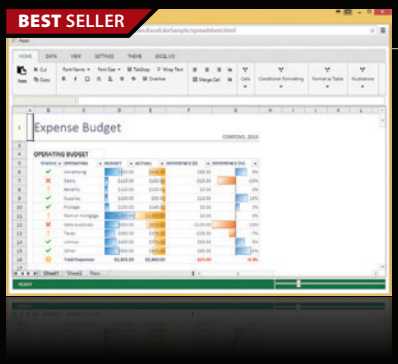


Apptimized | from $5,292.00

apptimized® APPLICATION PACKAGING

Discover, package, test and manage applications in the cloud.

- Rapidly test and prepare apps for Windows 10 without the need for hardware or software
- Capture a default install automatically and create customized discovery documentation
- The service is configured on a tenant philosophy, meaning that customer data is protected
- The advanced MSI Editor provides full MSI and App-V editing capability
- Supports bulk upload source media, and zip file capability for requests that have multiple EXEs



SpreadJS | from $1,476.51

GrapeCity

Deliver intuitive, efficient, multi-functional, pure JavaScript spreadsheets for Enterprise apps.

- Harness the power of a spreadsheet to display and manage data like Microsoft Excel
- Go beyond the grid with cards, trellis, calendar, Gantt, news feed, timeline and more
- Renders to the HTML canvas for a fast, interactive user experience across all browsers
- Modularized - so you only need to load the JavaScript that contains the features you need
- A Client-side component - works with Windows, Linux, MacOS, Android and iOS

Help & Manual Professional | from $586.04

ec software

Help and documentation for .NET and mobile applications.

- 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 responsive HTML, CHM, PDF, MS Word, ePUB, Kindle or print
- Styles and Templates give you full design control

DevExpress DXperience 17.1 | from $1,439.99

DevExpress

The complete range of DevExpress .NET controls and libraries for all major Microsoft platforms.

- WinForms - New TreeMap control, Chart series types and Unbound Data Source
- WPF - New Wizard control and Data Grid scrollbar annotations
- ASP.NET - New Vertical Grid control, additional Themes, Rich Editor Spell Checking and more
- Windows 10 Apps - New Hamburger Sub Menus, Splash Screen and Context Toolbar controls
- CodeRush - New debug visualizer expression map and code analysis diagnostics

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
2 New Century Place
East Street
Reading, Berkshire
RG1 4ET
United Kingdom

Asia / Pacific Headquarters
ComponentSource
7F Kojimachi Ichihara Bldg
1-1-8 Hirakawa-cho
Chiyoda-ku
Tokyo, 102-0093
Japan

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

mastercard AMERICAN EXPRESS VISA DISCOVER NETWORK

GSA Schedule Contract GS-35F-0188R

Figure 7 **A Second Data Point Added to the iris_sklearn.py Jobs Tab Charts**

If you refer back to **Figure 4**, you can see toward the lower middle of the screenshot that the predicted class is ['Iris-setosa'].

Passing Parameters

You may have noticed the Arguments textbox next to the Run button. Earlier, I left this field blank.

In the iris_sklearn.py file, lines 47 and 48, check for the presence of a parameter, convert the value to a float, and then set the reg variable to its value. If no parameters are sent to the program, the variable retains the value it was initialized with on line 45: 0.01, like so:

```
if len(sys.argv) > 1:
  reg = float(sys.argv[1])
```

> Tracked jobs have the added benefit of exploring the results graphically to make data experimentation faster.

The reg value gets passed as a parameter to the LogisticRegression method and sets the Regularization rate. Regularization rate controls the introduction of additional information in order to avoid overfitting the model. Overfitting occurs when the model performs too well on test data. A model that provides highly accurate results with test data will likely be unusable when given data outside of the test set in the future. More information about regularization and overfitting is available at bit.ly/2kfLU1f and bit.ly/2iatJpC, respectively.

Enter the number 10 into the Arguments textbox and click Run once more, making sure that Local and iris_sklearn.py are both selected. When completed, click on the Jobs tab, browse through it and choose iris_sklearn.py. Note that the charts now have a second data point, as shown in **Figure 7**.

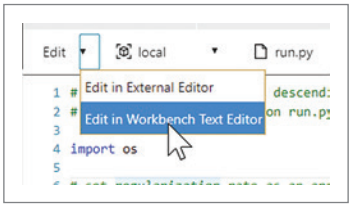


Figure 8 **Switch to Edit Mode**

Now, click on the files icon and click on the run.py file. This program will call the iris_sklearn.py file with a new regularization parameter that's half the value of the previous run until the value gets below 0.005. Because I already ran the script with a parameter of 10, I would like to change line seven to five. However, as you may have already noticed, the text isn't editable. To edit the text, click on the dropdown on the upper-left portion of the text area to switch to edit mode. There's an option to edit in another program, such as Visual Studio Code. However, in this case, choose Edit in Workbench Text Editor (see **Figure 8**).

After changing line 7 to reg = 5, click on the save icon. Next, click on Open Command Prompt from the File menu and type the following command and hit enter:

```
python run.py
```

The program will run and then pass a command to the underlying OS to run the iris_sklearn.py file through the Azure Azure ML Workbench system. Notice the program output in the command-line window and inside the Jobs pane within the Azure Azure ML Workbench program:

```
os.system('az ml experiment submit -c local ./iris_sklearn.py {}'.format(reg))
```

Click on the Jobs icon in the toolbar on the left-hand side of the screen and then click the entry for iris_sklearn.py. Notice how many more data points there are in the graphs. The run.py program executed the script a number of times with a different regularization rate each time.

Wrapping Up

In this article, I explored a common sample data set in data science with Azure ML Workbench, demonstrating the power and flexibility of the program. While not as straightforward or approachable as Azure ML Studio, Azure ML Workbench opens up a lot more possibilities to the data scientist and ML practitioner. First and foremost is the ability to install and consume any Python library, including Pickle, matplotlib and, of course, Sci-Kit Learn. There's also a command-line interface that can accept commands to install Python libraries and run Python code as tracked jobs inside Azure ML Workbench. Tracked jobs have the added benefit of exploring the results graphically to make data experimentation faster.

Azure ML Workbench includes several more features that I will explore in future articles, such as support for Jupyter notebooks, source control integration and Docker images. The tool truly brings great capabilities to the data science field. ■

Frank La Vigne *leads the Data & Analytics practice at Wintellect and co-hosts the DataDriven podcast. He blogs regularly at FranksWorld.com and you can watch him on his YouTube channel, "Frank's World TV" (FranksWorld.TV).*

Thanks to the following technical expert for reviewing this article: *Andy Leonard*

msdn magazine

MSDN MAGAZINE VENDOR PROFILE

RAVENDB

RavenDB OLTP Made Simple

RavenDB is one of the few fully transactional NoSQL databases. It's open source, speed-obsessed, and a pleasure to use.

What Makes RavenDB Unique

We developed RavenDB 4.0 to tackle the most common challenges your database faces in handling the scale and scope of today's data. Our credo of it just works strives to save your DBA and development team time and resources by resolving the most common issues before they occur, so they can devote more energy towards useful work.

Enjoy the Best of All Worlds

RavenDB employs the latest in database development to benefit your business in several ways:

High Speed Performance. On commodity servers, RavenDB can reach over 100,000 writes and half a million reads per second per node. Execute at millisecond performance right until you hit the limits of your hardware.

Easy to Install. You can install, set up, and secure RavenDB in a matter of minutes. The ramp up to learn is very quick and our SQL like query language makes RavenDB easy to use.

ACID. Upgrade to NoSql and take the best part of a relational database with you. Make your data integrity a constant with one of the only document databases that is fully transactional.

Scale up Quickly. Be ready for high traffic days like Black Friday by adding nodes to your database cluster and replicating your database to them in a matter of hours. Reduce latency and boost performance with additional capacity.

Flexibility. Our agile database is schemaless, giving you maximum flexibility. You don't have to fight your schema every time you want to make big changes. There is no need for expensive migrations. RavenDB lets you adapt quickly and seamlessly as you rise up the learning curve.

Keep Your Release Cycle Moving with Fewer Interruptions. We provide the best support in the industry, but we developed RavenDB so you won't need it. We want you to use RavenDB to build an application, not to waste precious time and money on the phone with our support engineers. RavenDB comes with a series of built-in fixes that help you cut down on overhead by anticipating the most common problems you can expect to encounter, and resolving them before they become an issue.

Always be there for your users. Our distributed database lets you operate over a cluster of servers, giving you high-availability to wherever your users are. Instant replication keeps several copies of your database running at once so your customers never have to wait in line.

Choose your own hosting. You can run RavenDB on-premise, or in the cloud through AWS or Azure.

Get Started at No Cost

Download RavenDB at **https://ravendb.net/get-ravendb-for-free** and try it out. Your community license includes 3 cores, a data cluster with up to 3 nodes, our state of the art GUI interface, and 6 GB of RAM.

About Hibernating Rhinos

Hibernating Rhinos LTD. is a leader in open source NoSQL database architecture and design. It's founder Oren Eini and his team are the creators of RavenDB, a leading OLTP document database that is fully transactional (ACID). Hibernating Rhinos operates in the United States, Europe, Latin America, and Israel. It's customer base includes Fortune 500 Companies, one of them running over 1.5 million instances of RavenDB.

For more information, visit → **https://ravendb.net/**

All About Span: Exploring a New .NET Mainstay

Stephen Toub

Imagine you're exposing a specialized sort routine to operate in-place on data in memory. You'd likely expose a method that takes an array and provide an implementation that operates over that T[]. That's great if your method's caller has an array and wants the whole array sorted, but what if the caller only wants part of it sorted? You'd probably then also expose an overload that took an offset and a count. But what if you wanted to support data in memory that wasn't in an array, but instead came from native code, for example, or lived on the stack and you only had a pointer and a length? How could you write your sort method that operated on such an arbitrary region of memory, and yet worked equally well with full arrays or with subsets of arrays, and that *also* worked equally well with managed arrays and unmanaged pointers?

Or take another example. You're implementing an operation over System.String, such as a specialized parsing method. You'd likely expose a method that takes a string and provide an implementation that operates on strings. But what if you wanted to support operating over a subset of that string? String.Substring could be used to carve out just the piece that's interesting to them, but that's a relatively expensive operation, involving a string allocation and memory copy. You could, as mentioned in the array example, take an offset and a count, but then what if the caller doesn't have a string but instead has a char[]? Or what if the caller has a char*, like one they created with stackalloc to use some space on the stack, or as the result of a call to native code? How could you write your parsing method in a way that didn't force the caller to do any allocations or copies, and yet worked equally well with inputs of type string, char[] and char*?

In both situations, you might be able to use unsafe code and pointers, exposing an implementation that accepted a pointer and a length. That, however, eliminates the safety guarantees that are core to .NET and opens you up to problems like buffer overruns and access violations that for most .NET developers are a thing of the past. It also invites additional performance penalties, such as needing to pin managed objects for the duration of the operation so that the pointer you retrieve remains valid. And depending on the type of data involved, getting a pointer at all may not be practical.

There's an answer to this conundrum, and its name is Span<T>.

This article discusses:

- New Span<T> and related types in .NET for safe and efficient memory access
- New C# language features for working with spans
- .NET runtime optimizations when working with spans

Technologies discussed:

Microsoft .NET Framework, .NET Core, C#

What Is Span<T>?

System.Span<T> is a new value type at the heart of .NET. It enables the representation of contiguous regions of arbitrary memory,

regardless of whether that memory is associated with a managed object, is provided by native code via interop, or is on the stack. And it does so while still providing safe access with performance characteristics like that of arrays.

For example, you can create a Span<T> from an array:

```
var arr = new byte[10];
Span<byte> bytes = arr; // Implicit cast from T[] to Span<T>
```

From there, you can easily and efficiently create a span to represent/point to just a subset of this array, utilizing an overload of the span's Slice method. From there you can index into the resulting span to write and read data in the relevant portion of the original array:

```
Span<byte> slicedBytes = bytes.Slice(start: 5, length: 2);
slicedBytes[0] = 42;
slicedBytes[1] = 43;
Assert.Equal(42, slicedBytes[0]);
Assert.Equal(43, slicedBytes[1]);
Assert.Equal(arr[5], slicedBytes[0]);
Assert.Equal(arr[6], slicedBytes[1]);
slicedBytes[2] = 44; // Throws IndexOutOfRangeException
bytes[2] = 45; // OK
Assert.Equal(arr[2], bytes[2]);
Assert.Equal(45, arr[2]);
```

As mentioned, spans are more than just a way to access and subset arrays. They can also be used to refer to data on the stack. For example:

```
Span<byte> bytes = stackalloc byte[2]; // Using C# 7.2 stackalloc support for spans
bytes[0] = 42;
bytes[1] = 43;
Assert.Equal(42, bytes[0]);
Assert.Equal(43, bytes[1]);
bytes[2] = 44; // throws IndexOutOfRangeException
```

More generally, they can be used to refer to arbitrary pointers and lengths, such as to memory allocated from a native heap, like so:

```
IntPtr ptr = Marshal.AllocHGlobal(1);
try
{
  Span<byte> bytes;
  unsafe { bytes = new Span<byte>((byte*)ptr, 1); }
  bytes[0] = 42;
  Assert.Equal(42, bytes[0]);
  Assert.Equal(Marshal.ReadByte(ptr), bytes[0]);
  bytes[1] = 43; // Throws IndexOutOfRangeException
}
finally { Marshal.FreeHGlobal(ptr); }
```

The Span<T> indexer takes advantage of a C# language feature introduced in C# 7.0 called ref returns. The indexer is declared with a "ref T" return type, which provides semantics like that of indexing into arrays, returning a reference to the actual storage location rather than returning a copy of what lives at that location:

```
public ref T this[int index] { get { ... } }
```

The impact of this ref-returning indexer is most obvious via example, such as by comparing it with the List<T> indexer, which is not ref returning. Here's an example:

```
struct MutableStruct { public int Value; }
...
Span<MutableStruct> spanOfStructs = new MutableStruct[1];
spanOfStructs[0].Value = 42;
Assert.Equal(42, spanOfStructs[0].Value);

var listOfStructs = new List<MutableStruct> { new MutableStruct() };
listOfStructs[0].Value = 42; // Error CS1612: the return value is not a variable
```

A second variant of Span<T>, called System.ReadOnlySpan<T>, enables read-only access. This type is just like Span<T>, except its indexer takes advantage of a new C# 7.2 feature to return a "ref readonly T" instead of a "ref T," enabling it to work with immutable data types like System.String. ReadOnlySpan<T> makes it very efficient to slice strings without allocating or copying, as shown here:

```
string str = "hello, world";
string worldString = str.Substring(startIndex: 7, length: 5); // Allocates
ReadOnlySpan<char> worldSpan =
  str.AsReadOnlySpan().Slice(start: 7, length: 5); // No allocation
Assert.Equal('w', worldSpan[0]);
worldSpan[0] = 'a'; // Error CS0200: indexer cannot be assigned to
```

Spans provide a multitude of benefits beyond those already mentioned. For example, spans support the notion of reinterpret casts, meaning you can cast a Span<byte> to be a Span<int> (where the 0th index into the Span<int> maps to the first four bytes of the Span<byte>). That way if you read a buffer of bytes, you can pass it off to methods that operate on grouped bytes as ints safely and efficiently.

How Is Span<T> Implemented?

Developers generally don't need to understand how a library they're using is implemented. However, in the case of Span<T>, it's worthwhile to have at least a basic understanding of the details behind it, as those details imply something about both its performance and its usage constraints.

> System.Span<T> is a new value type at the heart of .NET.

First, Span<T> is a value type containing a ref and a length, defined approximately as follows:

```
public readonly ref struct Span<T>
{
  private readonly ref T _pointer;
  private readonly int _length;
  ...
}
```

The concept of a ref T field may be strange at first—in fact, one can't actually declare a ref T field in C# or even in MSIL. But Span<T> is actually written to use a special internal type in the runtime that's treated as a just-in-time (JIT) intrinsic, with the JIT generating for it the equivalent of a ref T field. Consider a ref usage that's likely much more familiar:

```
public static void AddOne(ref int value) => value += 1;
...
var values = new int[] { 42, 84, 126 };
AddOne(ref values[2]);
Assert.Equal(127, values[2]);
```

This code passes a slot in the array by reference, such that (optimizations aside) you have a ref T on the stack. The ref T in the Span<T> is the same idea, simply encapsulated inside a struct. Types that contain such refs directly or indirectly are called ref-like types, and the C# 7.2 compiler allows declaration of such ref-like types by using ref struct in the signature.

From this brief description, two things should be clear:

1. Span<T> is defined in such a way that operations can be as efficient as on arrays: indexing into a span doesn't require computation to determine the beginning from a pointer and its starting offset, as the ref field itself already encapsulates both. (By contrast, ArraySegment<T> has a separate offset field, making it more expensive both to index into and to pass around.)

2. The nature of Span<T> as a ref-like type brings with it some constraints due to its ref T field.

This second item has some interesting ramifications that result in .NET containing a second and related set of types, led by Memory<T>.

What Is Memory<T> and Why Do You Need It?

Span<T> is a ref-like type as it contains a ref field, and ref fields can refer not only to the beginning of objects like arrays, but also to the middle of them:

```
var arr = new byte[100];
Span<byte> interiorRef1 = arr.AsSpan().Slice(start: 20);
Span<byte> interiorRef2 = new Span<byte>(arr, 20, arr.Length - 20);
Span<byte> interiorRef3 =
  Span<byte>.DangerousCreate(arr, ref arr[20], arr.Length - 20);
```

These references are called interior pointers, and tracking them is a relatively expensive operation for the .NET runtime's garbage collector. As such, the runtime constrains these refs to only live on the stack, as it provides an implicit low limit on the number of interior pointers that might be in existence.

Further, Span<T> as previously shown is larger than the machine's word size, which means reading and writing a span is not an atomic operation. If multiple threads read and write a span's fields on the heap at the same time, there's a risk of "tearing." Imagine an already initialized span containing a valid reference and a corresponding _length of 50. One thread starts writing a new span over it and gets as far as writing the new _pointer value. Then, before it can set the corresponding _length to 20, a second thread reads the span, including the new _pointer but the old (and longer) _length.

As a result, Span<T> instances can only live on the stack, not on the heap. This means you can't box spans (and thus can't use Span<T> with existing reflection invoke APIs, for example, as they require boxing). It means you can't have Span<T> fields in classes, or even in non-ref-like structs. It means you can't use spans in places where they might implicitly become fields on classes, for instance by capturing them into lambdas or as locals in async methods or iterators (as those "locals" may end up being fields on the compiler-generated state machines.) It also means you can't use Span<T> as a generic argument, as instances of that type argument could end up getting boxed or otherwise stored to the heap (and there's currently no "where T : ref struct" constraint available).

These limitations are immaterial for many scenarios, in particular for compute-bound and synchronous processing functions. But asynchronous functionality is another story. Most of the issues cited at the beginning of this article around arrays, array slices, native memory, and so on exist whether dealing with synchronous or asynchronous operations. Yet, if Span<T> can't be stored to the heap and thus can't be persisted across asynchronous operations, what's the answer? Memory<T>.

Memory<T> looks very much like an ArraySegment<T>:

```
public readonly struct Memory<T>
{
  private readonly object _object;
  private readonly int _index;
  private readonly int _length;
  ...
}
```

You can create a Memory<T> from an array and slice it just as you would a span, but it's a (non-ref-like) struct and can live on the heap. Then, when you want to do synchronous processing, you can get a Span<T> from it, for example:

```
static async Task<int> ChecksumReadAsync(Memory<byte> buffer, Stream stream)
{
  int bytesRead = await stream.ReadAsync(buffer);
  return Checksum(buffer.Span.Slice(0, bytesRead));
  // Or buffer.Slice(0, bytesRead).Span
}

static int Checksum(Span<byte> buffer) { ... }
```

As with Span<T> and ReadOnlySpan<T>, Memory<T> has a read-only equivalent, ReadOnlyMemory<T>. And as you'd expect, its Span property returns a ReadOnlySpan<T>. See **Figure 1** for a quick summary of built-in mechanisms for converting between these types.

You'll notice that Memory<T>'s _object field isn't strongly typed as T[]; rather, it's stored as an object. This highlights that Memory<T> can wrap things other than arrays, like System.Buffers.OwnedMemory<T>. OwnedMemory<T> is an abstract class that can be used to wrap data that needs to have its lifetime tightly managed, such as memory retrieved from a pool. That's a more advanced topic beyond the scope of this article, but

Figure 1 **Non-Allocating/Non-Copying Conversions Between Span-Related Types**

From	To	Mechanism
ArraySegment<T>	Memory<T>	Implicit cast, AsMemory method
ArraySegment<T>	ReadOnlyMemory<T>	Implicit cast, AsReadOnlyMemory method
ArraySegment<T>	ReadOnlySpan<T>	Implicit cast, AsReadOnlySpan method
ArraySegment<T>	Span<T>	Implicit cast, AsSpan method
ArraySegment<T>	T[]	Array property
Memory<T>	ArraySegment<T>	TryGetArray method
Memory<T>	ReadOnlyMemory<T>	Implicit cast, AsReadOnlyMemory method
Memory<T>	Span<T>	Span property
ReadOnlyMemory<T>	ArraySegment<T>	DangerousTryGetArray method
ReadOnlyMemory<T>	ReadOnlySpan<T>	Span property
ReadOnlySpan<T>	ref readonly T	Indexer get accessor, marshaling methods
Span<T>	ReadOnlySpan<T>	Implicit cast, AsReadOnlySpan method
Span<T>	ref T	Indexer get accessor, marshaling methods
String	ReadOnlyMemory<char>	AsReadOnlyMemory method
String	ReadOnlySpan<char>	Implicit cast, AsReadOnlySpan method
T[]	ArraySegment<T>	Ctor, Implicit cast
T[]	Memory<T>	Ctor, Implicit cast, AsMemory method
T[]	ReadOnlyMemory<T>	Ctor, Implicit cast, AsReadOnlyMemory method
T[]	ReadOnlySpan<T>	Ctor, Implicit cast, AsReadOnlySpan method
T[]	Span<T>	Ctor, Implicit cast, AsSpan method
void*	ReadOnlySpan<T>	Ctor
void*	Span<T>	Ctor

it's how Memory<T> can be used to, for example, wrap pointers into native memory. ReadOnlyMemory<char> can also be used with strings, just as can ReadOnlySpan<char>.

How Do Span<T> and Memory<T> Integrate with .NET Libraries?

In the previous Memory<T> code snippet, you'll notice a call to Stream.ReadAsync that's passing in a Memory<byte>. But Stream.ReadAsync in .NET today is defined to accept a byte[]. How does that work?

> Hundreds of new members and types are being added across .NET.

In support of Span<T> and friends, hundreds of new members and types are being added across .NET. Many of these are overloads of existing array-based and string-based methods, while others are entirely new types focused on specific areas of processing. For example, all the primitive types like Int32 now have Parse overloads that accept a ReadOnlySpan<char> in addition to the existing overloads that take strings. Imagine a situation where you're expecting a string that contains two numbers separated by a comma (such as "123,456"), and you want to parse out those two numbers. Today you might write code like this:

```
string input = ...;
int commaPos = input.IndexOf(',');
int first = int.Parse(input.Substring(0, commaPos));
int second = int.Parse(input.Substring(commaPos + 1));
```

That, however, incurs two string allocations. If you're writing performance-sensitive code, that may be two string allocations too many. Instead, you can now write this:

```
string input = ...;
ReadOnlySpan<char> inputSpan = input.AsReadOnlySpan();
int commaPos = input.IndexOf(',');
int first = int.Parse(inputSpan.Slice(0, commaPos));
int second = int.Parse(inputSpan.Slice(commaPos + 1));
```

By using the new Span-based Parse overloads, you've made this whole operation allocation-free. Similar parsing and formatting methods exist for primitives like Int32 up through core types like DateTime, TimeSpan and Guid, and even up to higher-level types like BigInteger and IPAddress.

In fact, many such methods have been added across the framework. From System.Random to System.Text.StringBuilder to System.Net.Sockets, overloads have been added to make working with {ReadOnly}Span<T> and {ReadOnly}Memory<T> simple and efficient. Some of these even carry with them additional benefits. For example, Stream now has this method:

```
public virtual ValueTask<int> ReadAsync(
  Memory<byte> destination,
  CancellationToken cancellationToken = default) { ... }
```

You'll notice that unlike the existing ReadAsync method that accepts a byte[] and returns a Task<int>, this overload not only accepts a Memory<byte> instead of a byte[], but also returns a ValueTask<int> instead of a Task<int>. ValueTask<T> is a struct that helps avoid allocations in cases where an asynchronous method is frequently expected to return synchronously, and where it's unlikely we can cache a completed task for all common return values. For instance, the runtime can cache a completed Task<bool> for a result of true and one for a result of false, but it can't cache four billion task objects for all possible result values of a Task<int>.

Because it's quite common for Stream implementations to buffer in a way that makes ReadAsync calls complete synchronously, this new ReadAsync overload returns a ValueTask<int>. This means asynchronous Stream read operations that complete synchronously can be entirely allocation-free. ValueTask<T> is also used in other new overloads, such as in overloads of Socket.ReceiveAsync, Socket.SendAsync, WebSocket.ReceiveAsync and TextReader.ReadAsync.

In addition, there are places where Span<T> allows the framework to include methods that in the past raised memory safety concerns. Consider a situation where you want to create a string containing a randomly generated value, such as for an ID of some kind. Today you might write code that requires allocating a char array, like this:

```
int length = ...;
Random rand = ...;
var chars = new char[length];
for (int i = 0; i < chars.Length; i++)
{
  chars[i] = (char)(rand.Next(0, 10) + '0');
}
string id = new string(chars);
```

You could instead use stack-allocation, and even take advantage of Span<char>, to avoid needing to use unsafe code. This approach also takes advantage of the new string constructor that accepts a ReadOnlySpan<char>, like so:

```
int length = ...;
Random rand = ...;
Span<char> chars = stackalloc char[length];
for (int i = 0; i < chars.Length; i++)
{
  chars[i] = (char)(rand.Next(0, 10) + '0');
}
string id = new string(chars);
```

This is better, in that you've avoided the heap allocation, but you're still forced to copy into the string the data that was generated on the stack. This approach also only works when the amount of space required is something small enough for the stack. If the length is short, like 32 bytes, that's fine, but if it's thousands of bytes, it could easily lead to a stack overflow situation. What if you could write to the string's memory directly instead? Span<T> allows you to do that. In addition to string's new constructor, string now also has a Create method:

```
public static string Create<TState>(
  int length, TState state, SpanAction<char, TState> action);
...
public delegate void SpanAction<T, in TArg>(Span<T> span, TArg arg);
```

This method is implemented to allocate the string and then hand out a writable span you can write to in order to fill in the contents of the string while it's being constructed. Note that the stack-only nature of Span<T> is beneficial in this case, guaranteeing that the span (which refers to the string's internal storage) will cease to exist before the string's constructor completes, making it

impossible to use the span to mutate the string after the construction is complete:

```
int length = ...;
Random rand = ...;
string id = string.Create(length, rand, (Span<char> chars, Random r) =>
{
  for (int i = 0; chars.Length; i++)
  {
    chars[i] = (char)(r.Next(0, 10) + '0');
  }
});
```

Now, not only have you avoided the allocation, you're writing directly into the string's memory on the heap, which means you're also avoiding the copy and you're not constrained by size limitations of the stack.

> The runtime can apply for spans the same kinds of optimizations it does for arrays, making spans efficient for accessing data.

Beyond core framework types gaining new members, many new .NET types are being developed to work with spans for efficient processing in specific scenarios. For example, developers looking to write high-performance microservices and Web sites heavy in text processing can earn a significant performance win if they don't have to encode to and decode from strings when working in UTF-8. To enable this, new types like System.Buffers.Text.Base64, System.Buffers.Text.Utf8Parser and System.Buffers.Text.Utf8Formatter are being added. These operate on spans of bytes, which not only avoids the Unicode encoding and decoding, but enables them to work with native buffers that are common in the very lowest levels of various networking stacks:

```
ReadOnlySpan<byte> utf8Text = ...;
if (!Utf8Parser.TryParse(utf8Text, out Guid value,
  out int bytesConsumed, standardFormat = 'P'))
  throw new InvalidDataException();
```

All this functionality isn't just for public consumption; rather the framework itself is able to utilize these new Span<T>-based and Memory<T>-based methods for better performance. Call sites across .NET Core have switched to using the new ReadAsync overloads to avoid unnecessary allocations. Parsing that had been done by allocating substrings now takes advantage of allocation-free parsing. Even niche types like Rfc2898DeriveBytes have gotten in on the action, taking advantage of the new Span<byte>-based TryComputeHash method on System.Security.Cryptography.HashAlgorithm to achieve a monstrous savings on allocation (a byte array per iteration of the algorithm, which might iterate thousands of times), as well as a throughput improvement.

This doesn't stop at the level of the core .NET libraries; it continues all the way up the stack. ASP.NET Core now has a heavy dependency on spans, for example, with the Kestrel server's HTTP parser written on top of them. In the future, it's likely that spans will be exposed out of public APIs in the lower levels of ASP.NET Core, such as in its middleware pipeline.

What About the .NET Runtime?

One of the ways the .NET runtime provides safety is by ensuring that indexing into an array doesn't allow going beyond the length of the array, a practice known as bounds checking. For example, consider this method:

```
[MethodImpl(MethodImplOptions.NoInlining)]
static int Return4th(int[] data) => data[3];
```

On the x64 machine on which I'm typing this article, the generated assembly for this method looks like the following:

```
       sub     rsp, 40
       cmp     dword ptr [rcx+8], 3
       jbe     SHORT G_M22714_IG04
       mov     eax, dword ptr [rcx+28]
       add     rsp, 40
       ret
G_M22714_IG04:
       call    CORINFO_HELP_RNGCHKFAIL
       int3
```

That cmp instruction is comparing the length of the data array against the index 3, and the subsequent jbe instruction is then jumping to the range check failure routine if 3 is out of range (for an exception to be thrown). The JIT needs to generate code that ensures such accesses don't go outside the bounds of the array, but that doesn't mean that every individual array access needs a bound check. Consider this Sum method:

```
static int Sum(int[] data)
{
  int sum = 0;
  for (int i = 0; i < data.Length; i++) sum += data[i];
  return sum;
}
```

The JIT needs to generate code here that ensures the accesses to data[i] don't go outside the bounds of the array, but because the JIT can tell from the structure of the loop that i will always be in range (the loop iterates through each element from beginning to end), the JIT can optimize away the bounds checks on the array. Thus, the assembly code generated for the loop looks like the following:

```
G_M33811_IG03:
       movsxd  r9, edx
       add     eax, dword ptr [rcx+4*r9+16]
       inc     edx
       cmp     r8d, edx
       jg      SHORT G_M33811_IG03
```

A cmp instruction is still in the loop, but simply to compare the value of i (as stored in the edx register) against the length of the array (as stored in the r8d register); no additional bounds checking.

The runtime applies similar optimizations to span (both Span<T> and ReadOnlySpan<T>). Compare the previous example to the following code, where the only change is on the parameter type:

```
static int Sum(Span<int> data)
{
  int sum = 0;
  for (int i = 0; i < data.Length; i++) sum += data[i];
  return sum;
}
```

The generated assembly for this code is almost identical:

```
G_M33812_IG03:
       movsxd  r9, r8d
       add     ecx, dword ptr [rax+4*r9]
       inc     r8d
       cmp     r8d, edx
       jl      SHORT G_M33812_IG03
```

The assembly code is so similar in part because of the elimination of bounds checks. But also relevant is the JIT's recognition of the span indexer as an intrinsic, meaning that the JIT generates

special code for the indexer, rather than translating its actual IL code into assembly.

All of this is to illustrate that the runtime can apply for spans the same kinds of optimizations it does for arrays, making spans an efficient mechanism for accessing data. More details are available in the blog post at bit.ly/2zywvyl.

What About the C# Language and Compiler?

I've already alluded to features added to the C# language and compiler to help make Span<T> a first-class citizen in .NET. Several features of C# 7.2 are related to spans (and in fact the C# 7.2 compiler will be required to use Span<T>). Let's look at three such features.

> Several features have been added to the C# language and compiler to help make Span<T> a first-class citizen in .NET.

Ref structs. As noted earlier, Span<T> is a ref-like type, which is exposed in C# as of version 7.2 as ref struct. By putting the ref keyword before struct, you tell the C# compiler to allow you to use other ref struct types like Span<T> as fields, and in doing so also sign up for the associated constraints to be assigned to your type. For example, if you wanted to write a struct Enumerator for a Span<T>, that Enumerator would need to store the Span<T> and, thus, would itself need to be a ref struct, like this:

```
public ref struct Enumerator
{
  private readonly Span<char> _span;
  private int _index;
  ...
}
```

Stackalloc initialization of spans. In previous versions of C#, the result of stackalloc could only be stored into a pointer local variable. As of C# 7.2, stackalloc can now be used as part of an expression and can target a span, and that can be done without using the unsafe keyword. Thus, instead of writing:

```
Span<byte> bytes;
unsafe
{
  byte* tmp = stackalloc byte[length];
  bytes = new Span<byte>(tmp, length);
}
```

You can write simply:

```
Span<byte> bytes = stackalloc byte[length];
```

This is also extremely useful in situations where you need some scratch space to perform an operation, but want to avoid allocating heap memory for relatively small sizes. Previously you had two choices:

- Write two completely different code paths, allocating and operating over stack-based memory and over heap-based memory.
- Pin the memory associated with the managed allocation and then delegate to an implementation also used for the stack-based memory and written with pointer manipulation in unsafe code.

Now, the same thing can be accomplished without code duplication, with safe code and with minimal ceremony:

```
Span<byte> bytes = length <= 128 ? stackalloc byte[length] : new byte[length];
... // Code that operates on the Span<byte>
```

Span usage validation. Because spans can refer to data that might be associated with a given stack frame, it can be dangerous to pass spans around in a way that might enable referring to memory that's no longer valid. For example, imagine a method that tried to do the following:

```
static Span<char> FormatGuid(Guid guid)
{
  Span<char> chars = stackalloc char[100];
  bool formatted = guid.TryFormat(chars, out int charsWritten, "d");
  Debug.Assert(formatted);
  return chars.Slice(0, charsWritten); // Uh oh
}
```

Here space is being allocated from the stack and then trying to return a reference to that space, but the moment you return, that space will no longer be valid for use. Thankfully the C# compiler detects such invalid usage with ref structs and fails the compilation with an error:

```
error CS8352: Cannot use local 'chars' in this context because it may
expose referenced variables outside of their declaration scope
```

What's Next?

The types, methods, runtime optimizations, and other elements discussed here are on track to being included in .NET Core 2.1. After that, I expect them to make their way into the .NET Framework. The core types like Span<T>, as well as the new types like Utf8Parser, are also on track to being made available in a System.Memory.dll package that's compatible with .NET Standard 1.1. That will make the functionality available for existing releases of .NET Framework and .NET Core, albeit without some of the optimizations implemented when built into to the platform. A preview of this package is available for you to try out today—simply add a reference to the System.Memory.dll package from NuGet.

Of course, keep in mind that there can and will be breaking changes between the current preview version and what's actually delivered in a stable release. Such changes will in large part be due to feedback from developers like you as you experiment with the feature set. So please do give it a try, and keep an eye on the github.com/dotnet/coreclr and github.com/dotnet/corefx repositories for ongoing work. You can also find documentation at aka.ms/ref72.

Ultimately, the success of this feature set relies on developers trying it out, providing feedback, and building their own libraries utilizing these types, all with the goal of providing efficient and safe access to memory in modern .NET programs. We look forward to hearing from you about your experiences, and even better, to working with you on GitHub to improve .NET further. ■

Stephen Toub *works on .NET at Microsoft. You can find him on GitHub at github.com/stephentoub.*

Thanks to the following technical experts for reviewing this article: *Krzysztof Cwalina, Eric Erhardt, Ahson Khan, Jan Kotas, Jared Parsons, Marek Safar, Vladimir Sadov, Joseph Tremoulet, Bill Wagner, Jan Vorlicek, Karel Zikmund*

Visual Studio LIVE!

EXPERT SOLUTIONS FOR .NET DEVELOPERS

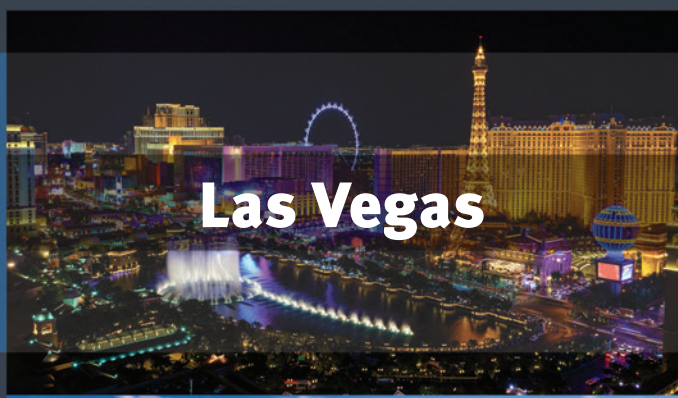
Visual Studio LIVE! 25 YEARS OF CODING INNOVATION

VISUAL STUDIO LIVE! (VSLive!™) is celebrating 25 years as one of the most respected, longest-standing, independent developer conferences, and we want you to be a part of it.

Join us in 2018 for #VSLive25, as we highlight how far technology has come in 25 years, while looking toward the future with our unique brand of training on .NET, the Microsoft Platform and open source technologies in seven great cities across the US.

March 12 – 16, 2018
Bally's Hotel & Casino

Respect the Past. Code the Future.



April 30 – May 4, 2018
Hyatt Regency Austin

Code Like It's 2018!

June 10 – 14, 2018
Hyatt Regency Cambridge

Developing Perspective.

SUPPORTED BY

Microsoft | Visual Studio | msdn magazine | Visual Studio MAGAZINE

PRODUCED BY

1105 MEDIA
YOUR GROWTH. OUR BUSINESS.

Visual Studio LIVE!

2018 DATES ANNOUNCED

August 13 – 17, 2018
Microsoft Headquarters

Yesterday's Knowledge; Tomorrow's Code!

September 17 – 20, 2018
Renaissance Chicago

Look Back to Code Forward.

NEW LOCATION!

October 8 – 11, 2018
Hilton San Diego Resort

Code Again for the First Time!

December 2 – 7, 2018
Loews Royal Pacific Resort

Code Odyssey.

CONNECT WITH US

twitter.com/vslive – @VSLive

facebook.com – Search "VSLive"

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

vslive.com

#VSLIVE25

Extend Excel Formulas for Data Analysis

Michael Saunders

Take a second to think about Excel as a whole—it's a flexible, powerful, ubiquitous tool for analyzing data, from my handy little to-do list spreadsheet, all the way to massive 100MB financial-model workbooks at the world's top banks. And underneath all the fancy features, behind the slick charts and formatting, the real value is calculation, called "Calc" internally. Calc is what lets you create relationships between cells: You express complex models by writing simple formulas to describe the dependency trees between values (see **Figure 1**). And then, as soon as you make a change, Calc updates the dependent values based on those formulas.

Excel provides tons of helpful functions to use in your formulas, from the essential AVERAGE function (bit.ly/1Tjynwl), to string-analysis functions like SEARCH for finding substrings (bit.ly/2jhcEuV), to more complex calculations like the T.TEST statistical function (bit.ly/2ipowKE). People at work and in school learn Excel functions to use in formulas, and they interact with them naturally for their day-to-day calculation needs. So if you want to provide a powerful capability that Excel doesn't already have, consider building an Excel function for that purpose. See **Figure 2** for an example: The Contoso Cryptographers Corp. wants to release a solution that helps sleuths analyze data in Excel to decode messages. And one of the handy tools for code breaking is to recognize primes quickly, so Contoso would love to have an ISPRIME function for the millions of detectives who already use Excel. Later in this article you'll see how Contoso builds this function and the other capabilities it needs for its add-in.

Some of the technologies discussed in this article are in preview; all information is subject to change.

This article discusses:

- Excel add-ins
- JavaScript custom functions in Excel
- Azure Machine Learning functions in Excel

Technologies discussed:

Excel, JavaScript, Azure Machine Learning

What Can I Build?

Before you start building, consider that not all extensions to Excel functionality should use Excel formulas. To decide if yours should, ask yourself this: "Does my function need to change anything aside from the Excel value that's being returned?" In other words, does the function have side effects? In Excel, people don't expect formulas to change anything except the cell in which they're entered. That change might trigger other changes, but those would also be the result of self-contained formulas without side effects. When you type "=SUM()" in cell A1, you don't expect a chart to appear somewhere on the sheet or a new row to be added underneath or a number to be changed in a financial database on the Internet, though you *can* control all those behaviors as part of a solution you build, which might contain functions and other capabilities.

The type of solution you should build to create your Excel function depends on your skills and goals. If you're a professional developer, either selling externally like Contoso Cryptographers or building for people in your own organization, an add-in is the right tool for data analytics solutions in Excel, as is explained in the "JavaScript Custom Functions in Excel Add-Ins section." If you're an AI developer or data scientist looking to build intelligent models for data analysts to use at your company, check out the "Azure Machine Learning Functions" section later in this article. And if you don't have any of those skills, Microsoft will have solutions for you in the future!

JavaScript Custom Functions in Excel Add-Ins

Excel add-ins are tools that professional developers can build to extend Excel and interact with the data in workbooks (bit.ly/2AUOsZk). Thousands of add-ins are already available and they all run across Excel platforms, including Windows, Mac, Excel Online and even iOS. Anyone familiar with Web technologies will find add-ins easy to build, because they're written just like Web pages: Add-ins run HTML, JavaScript, and CSS, and have the ability to call external Web servers. Best of all, the add-ins don't need any code changes to run on all the different platforms. (Excel also supports other types

of add-ins, as described at bit.ly/2qsPfLe, but they can't run across platforms or be deployed to the Store.) **Figure 3** shows the key pieces of an add-in. The main piece is the manifest.xml file, which specifies the Web location of the rest of the add-in and other metadata. The rest of the code is deployed by the developer, just as with any other Web application.

Add-ins provide lots of useful ways to extend Excel (see the documentation at bit.ly/2AV47rw). They can add UI elements, like ribbon buttons, contextual menu options, and HTML task panes and dialogs. They can interact with workbooks via thousands of APIs, such as the abilities to add and remove rows, insert and edit charts, and even apply formatting to cells. And now, with the new preview release of JavaScript custom functions, it's possible to extend Excel formulas.

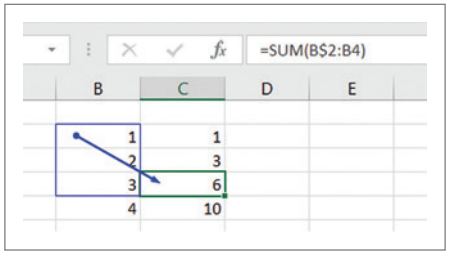


Figure 1 **Calc Lets You Create Relationships Between Cells**

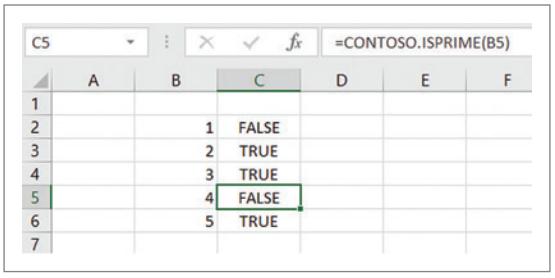


Figure 2 **The Contoso Function to Identify Prime Numbers**

Custom functions allow developers to add any JavaScript function to Excel using an add-in (bit.ly/2AYtNUW). Users can then access custom functions like any other native function in Excel (such as AVERAGE). Let's take a closer look at the Contoso Cryptographers ISPRIME function to see how it's written (It's also possible to check primality using only built-in Excel functions, but that's much more difficult for the person requesting the calculation.):

```
function isPrime(n) {
  var root = Math.sqrt(n);
  if (n < 2) return false;
  for (var divisor = 2; divisor <= root; divisor++){
    if(n % divisor == 0) return false;
  }
  return true;
}
```

The function simply checks all possible divisors up to the square root to determine whether the input is prime—a few lines of JavaScript. All the actual logic is done already. But there's more to write than just the function itself. To be an effective extension point, custom functions must look and feel just like native ones. And native functions have a bunch of customized information. Look at **Figure 4** for an example with the Contoso Crypographers ISPRIME function: In addition to the name of the function itself, Excel displays a helpful description to clarify the purpose of the function.

You can see another example in **Figure 5**. After CONTOSO.ISPRIME is selected, Excel shows the name of the parameter (in this case, n) to make it easier to pick the right inputs.

Both of these pieces of information and much more are provided by the developer in the JavaScript definition of the function's metadata, as you can see in **Figure 5**.

You can see that the description is specified as a string and so is the name of each parameter. I won't discuss all the metadata here, but you can see the documentation for more info. If you're familiar with the add-in model, you might be wondering why this information is provided in JavaScript rather than hardcoded statically somewhere, like in the manifest.xml file. The reason is flexibility. In Contoso's case, the cryptography functions are defined and well-known beforehand. But sometimes you might want the ability to enable different functions in different situations.

Contoso Cryptographers love the simplicity of their ISPRIME function, but their next goal is a little more difficult: They want to build a function to generate random numbers for encrypting text. Excel has an excellent RAND function, but the problem is that RAND isn't ideal for encryption because its values are pseudorandom, generated procedurally. In contrast, numbers generated by random.org are safe to use for this purpose—they're generated from atmospheric noise. Of course, it's no good to hardcode random numbers; instead, Contoso must design a function that can call random.org to fetch random numbers via HTTP request. Thankfully, custom functions make it easy to include Web requests. Here's what Contoso's asynchronous RANDOM function looks like:

```
function getRandom(min, max) {
  return new OfficeExtension.Promise(function(setResult, setError){
    sendRandomOrgHTTP(min, max, function(result){
      if(result.number) setResult(number);
      else setError(result.error);
    });
  });
}
```

The key difference between this function and ISPRIME is that RANDOM is asynchronous: instead of returning a value to Excel, it immediately returns a JavaScript promise, then makes an XMLHttpRequest (not visible in the function—it's in the sendRandomOrgHTTP method) to the random.org service. Then, once the Web service has responded, Contoso resolves the promise with the random.org number to write it to the cell. Bringing Web data into Excel is one of the key reasons companies want to extend Excel functions, so Microsoft expects asynchronous functions to be popular.

One common aspect of all asynchronous functions is that they take some amount of time to return their result, so Excel shows a #GETTING_DATA message in the cell while it waits for the function to be resolved (see **Figure 6**).

In this case, the HTTP request can complete quickly, in around a tenth of a second. But if necessary, the function can make many calls and take longer than that to evaluate. For JavaScript custom functions, resolving the value in JavaScript causes it to be displayed

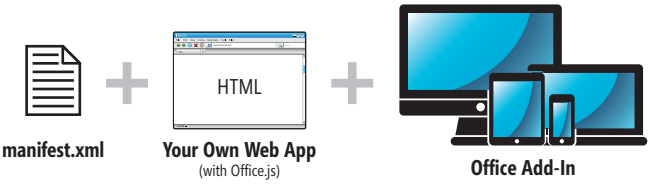


Figure 3 **The Key Pieces of an Add-In**

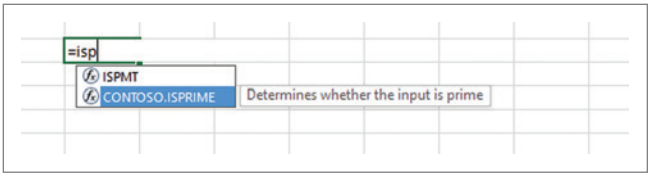


Figure 4 **Custom Functions and Native Functions Automatically Complete While You Type**

immediately in the cell. **Figure 7** shows an Excel view of the same RANDOM function after the value is returned.

Of course, an add-in using custom functions could have many other features, as well: Contoso Cryptographers might want to create a task pane and a ribbon tab to give their customers easy access to their custom functions; settings for how they behave; and guidance on how to use them. Perhaps the JavaScript API could even be used to let customers encrypt an entire worksheet of data with a click of a button. Whatever they decide, they can be confident that the entire add-in will run across Excel platforms without code changes.

There are two main ways to deploy an add-in containing these JavaScript custom functions, depending on the audience. The public Office Store (bit.ly/2A7OL5o) is available to anyone who has Excel—there's even a button to browse the Store on the Excel ribbon. The Store supports both free and paid add-ins, and requires submission and validation through Microsoft. However, Contoso Cryptographers Corp. deploys directly to its small business customers so they don't have to click anything to install it. Instead, it provides each customer with the manifest file. Then an IT admin can choose which users get access via the O365 admin center (see the interface in **Figure 8**). Those users get the add-in installed automatically.

Hopefully you've been inspired by the resourceful Contoso Cryptographers to try writing your own custom functions (use the guide and samples at aka.ms/customfunctions). Next, I'll explore the other new way to extend Excel formulas.

Azure Machine Learning Functions

The second type of extensible function Microsoft has announced is Azure Machine Learning functions (abbreviated Azure ML). In contrast to JavaScript custom functions, Azure ML functions are created by AI developers (often data scientists and other experts) for analysts in their organization to use. As a result, there's no need to create an add-in package to deploy an Azure ML function.

Figure 5 **The Contoso ISPRIME Function**

```
Excel.Script.CustomFunctions["CONTOSO"]["ISPRIME"] = {
  call: isPrime,
  description: "Determines whether the input is prime",
  helpUrl: "https://example.com/help.html",
  result: {
    resultType: Excel.CustomFunctionValueType.boolean,
    resultDimensionality: Excel.CustomFunctionDimensionality.scalar,
  },
  parameters: [{
    name: "n",
    description: "the number to be evaluated",
    valueType: Excel.CustomFunctionValueType.number,
    valueDimensionality: Excel.CustomFunctionDimensionality.scalar,
  }],
  options: { batched: false, streaming: false }
};
```

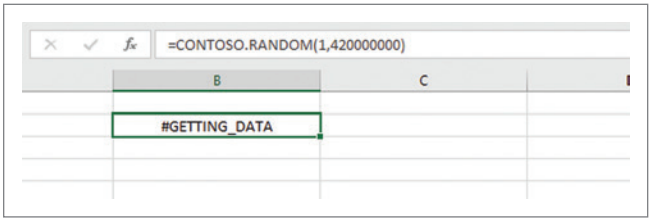


Figure 6 **Waiting for an Asynchronous Function to Return**

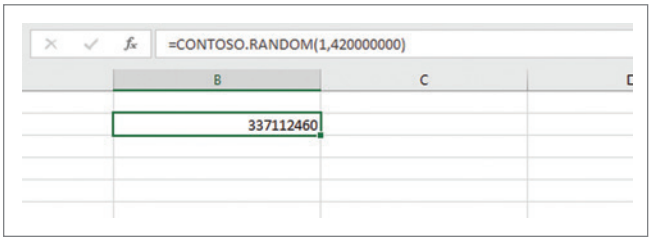


Figure 7 **An Excel View of the RANDOM Function After the Value Is Returned**

The Azure ML function itself is based on a service that calculates or predicts values based on a machine learning (ML) model. Once the model is built, the creators can enable it for anyone they choose. Then, every time someone wants to run the function, they simply type in a cell, just like for any other Excel function. The function calls a live Web service on the company's Azure subscription and returns the result asynchronously.

For example, a marketing analyst at a retailer might want to predict demand for new products in various geographies. The analyst has some data on that product and the target markets in Excel. The goal is to get a simple function that lets the analyst forecast demand in each market without ever leaving Excel. **Figure 9** shows a sample spreadsheet with this type of data.

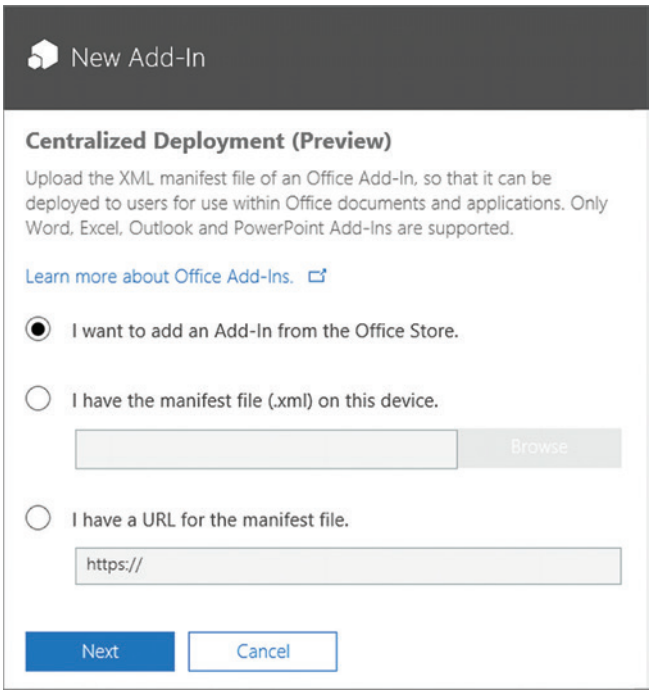


Figure 8 **Deploying a Custom Function to an Organization**

Visual Studio LIVE!

EXPERT SOLUTIONS FOR .NET DEVELOPERS

LAS VEGAS
MAR 11-16 2018
BALLY'S, LAS VEGAS, NV

Visual Studio LIVE! 25 YEARS OF CODING INNOVATION

RESPECT THE PAST; CODE THE FUTURE!

VSLive! 1998

VSLive! 2017

INTENSE TRAINING FOR DEVELOPERS, ENGINEERS, PROGRAMMERS AND ARCHITECTS:

- Visual Studio
- .NET Core
- Xamarin
- Software Practices
- Angular JS
- ASP.NET / Web Server
- Database and Analytics
- ALM / DevOps
- Cloud Computing
- UWP/Windows

- **BACK BY POPULAR DEMAND: Hands-On Labs!**
  Sunday, March 11.
  Starting at $645 thru January 19.

Register by January 19 and Save $400!*

Use promo code VSLTIPIN
*Available on 3, 5, and 6 day packages only.

REGISTER NOW

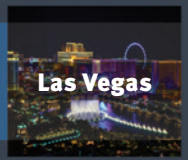
vslive.com/vegas

Visual Studio LIVE! 2018

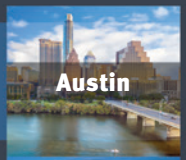
EXPERT SOLUTIONS FOR .NET DEVELOPERS

Help Us Celebrate #VSLive25! Which Location Will You Attend?

March 11 – 16, 2018
Bally's Hotel & Casino
Respect the Past. Code the Future.



April 30 – May 4, 2018
Hyatt Regency Austin
Code Like It's 2018!



June 10 – 14, 2018
Hyatt Regency Cambridge
Developing Perspective.

August 13 – 17, 2018
Microsoft Headquarters
Yesterday's Knowledge; Tomorrow's Code!

September 17 – 20, 2018
Renaissance Chicago
Look Back to Code Forward.

NEW LOCATION!
October 7 – 11, 2018
Hilton San Diego Resort
Code Again for the First Time!

December 2 – 7, 2018
Loews Royal Pacific Resort
Code Odyssey.

Visual Studio LIVE! 25 YEARS OF CODING INNOVATION

CONNECT WITH US

twitter.com/vslive – @VSLive

facebook.com – Search "VSLive"

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

EVENT PARTNER Microsoft

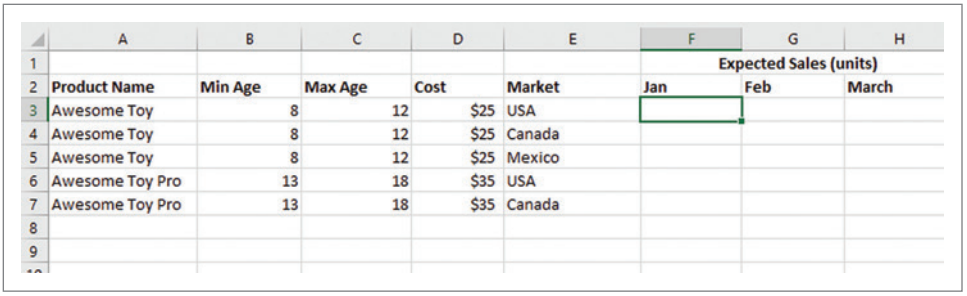
SUPPORTED BY Visual Studio msdn magazine Visual Studio MAGAZINE

PRODUCED BY 1105 MEDIA

vslive.com/vegas

An AI developer at that organization uses Azure ML services (bit.ly/2nwa0WP) to create the model: She starts by preparing training data, typically from an external database, to build an ML experiment based on historical sales for new products. Azure Machine Learning Workbench is a tool that simplifies the data preparation and lets the data scientist write Python code to train and evaluate models. **Figure 10** is the Run Dashboard view in Azure ML Workbench, showing statistics on an experiment.

Once the model is ready, it can be deployed as a Web service in the organization's Azure subscription, with permissions for the right set of users to access it. Instead of defining the metadata in JavaScript that runs locally, the metadata is defined in a Swagger (RESTful API) format that's created automatically along with the deployed service. All that's required for the service to appear as a function in Excel is that the analyst in Excel has permission to access it. **Figure 11** shows the deployed functions appearing for the data analysts in Excel. The analysts can then run the function normally. Just as with the asynchronous JavaScript custom functions, the cell shows a #GETTING_DATA message while the service computes the result.



	A	B	C	D	E	F	G	H
1						Expected Sales (units)		
2	Product Name	Min Age	Max Age	Cost	Market	Jan	Feb	March
3	Awesome Toy	8	12	$25	USA			
4	Awesome Toy	8	12	$25	Canada			
5	Awesome Toy	8	12	$25	Mexico			
6	Awesome Toy Pro	13	18	$35	USA			
7	Awesome Toy Pro	13	18	$35	Canada			
8								
9								

Figure 9 **Sample Product and Market Data That Will Be Analyzed with an Azure ML Function**

Figure 10 **Azure Machine Learning Workbench**

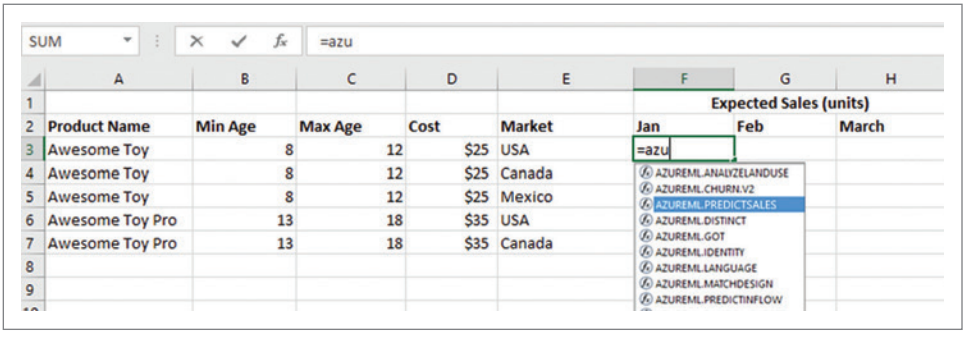


Figure 11 **The Deployed Functions Showing in Excel**

The Future of Extensible Functions in Excel

I hope you'll try the technology previews for extensible functions in Excel—JavaScript custom functions and add-ins if you're a software or services vendor, and Azure ML functions if you're an AI developer or data scientist. Microsoft is listening carefully to all the feedback during the preview (post ideas to the UserVoice page at bit.ly/2jRJQsu), so in addition to getting a head start, you'll also have the opportunity to shape the final released product. There are many more improvements coming to both of these function types, and some entirely new ones, so stay posted for future announcements! ■

Michael Saunders *is a program manager on the Office team, where he builds Excel features for developers. He's originally from Toronto, Canada, and he studied Management and Materials Science Engineering at the University of Pennsylvania and Wharton. In his spare time, Saunders sings with the Seattle Esoterics and develops add-ins.*

Thanks to the following Microsoft technical experts for reviewing this article: *Yina Arenas, Ashvini Sharma, Sandhya Vankamamidi*

Build the API to Your Organization with Microsoft Graph and Azure Functions

Mike Ammerlaan

If you think of your organization as an API, what would it look like?

You'd probably start with the people—the heart of an organization—and the kinds of roles and functions they fulfill. Such people are frequently grouped into defined and virtual teams that accomplish tasks and projects. You'd layer on resources, including where people work and the tools they use to get work done. You'd then add processes and work activities—perhaps these are methods in the API? Though "widgetMarketingTeam.runCampaign()" is perhaps wildly overly simplistic, nevertheless, with an API to your organization, you'd get great insights into how your organization is functioning and could transform productivity by building more efficient processes and tools.

The key is making every resource available consistently, and interconnected logically, so you can craft comprehensive processes to fit the way individuals and teams want to work. The more APIs you can bring together and connect, the more useful by far the net set of products you build could be—greater than the sum of its parts.

For this, we offer Microsoft Graph—an API that spans key data sets within your organization and allows you to pull together everything you need to transform how work is performed. Moreover, Microsoft Graph works with consumer services, such as OneDrive and Mail (Outlook.com), empowering you to transform personal productivity, as well.

This article discusses Azure Functions Binding Extensions, which is in public preview. All information is subject to change.

This article discusses:

- Using Microsoft Graph to unify data and authentication across organizational systems
- Building processes and integrations in Microsoft Graph using Azure Functions
- Creating a task using Azure Functions
- Processing files in OneDrive

Technologies discussed:

Microsoft Graph, Azure Functions, Azure Functions Bindings Extensions for Microsoft Graph, Cognitive Services Speech API

Solving the Problem of API Sprawl

Across an organization, the set of software systems in use can vary wildly. For developers, each presents a unique structure, typically with a distinct set of APIs, authentication requirements and style of interaction. Frequently, a major challenge of software projects is simply bridging these different systems to provide a higher level of insight and can include abstracting out different APIs and mastering individual authentication schemes.

Historically, individual APIs from different product teams—at Microsoft, in my case—would work differently and require cross-product integration. Even five years ago, the process of getting a user's complete profile and photo would require callouts

to both Exchange APIs (to get information about a person) and SharePoint APIs (to get a photo from a user's managed profile). Each had their own authentication, API scheme, and differing requirements. What if you then wanted to get the information about a person's manager? That would involve querying a third system to get organizational hierarchy. These operations were all *possible* to pull together, but more complex than they needed to be.

Microsoft Graph was born out of a desire to solve this problem. By unifying data and authentication, and making systems consistent, the net set of APIs becomes much easier and more practical to use. Microsoft Graph pulls together diverse systems from across your organization, representing key facets and functions in a company. Since its launch two years ago, Microsoft Graph has continued to grow in breadth of both functionality and capability, to where it really can serve as a foundational API for your organization.

At the core of Microsoft Graph is the set of users—typically all employees with an account in an organization. Simplified, centralized groups are an emerging concept in Microsoft Graph, generally starting with a list of users and other security groups. Groups can have an associated set of resources, like a Microsoft Teams chat-based workspace, a Planner task board, and a SharePoint site with document libraries and files. From there, various tools of work are represented for users and groups, including files via the Drive API, tasks via the Planner API, incoming mail for both users and groups, contacts, calendar, and more, as shown in **Figure 1**.

Over time, new capabilities have been added across the APIs in Microsoft Graph. A new ability to persist custom metadata along with items in Microsoft Graph gives you the ability to deeply customize these items. Now a group is no longer just a group—with additional metadata describing topic, instructor and timing, a group could represent a class in an educational institution. You could use this metadata to then perform queries—for example, find all groups that represent science classes. Alternatively, you could connect your systems into Microsoft Graph by adding identifiers from your system to the related entities within Microsoft Graph.

Microsoft Graph also goes beyond providing create, read, update and delete (CRUD) APIs for core objects. A major feature is a layer of insights that are generated behind the scenes as users work. For example, though Graph contains a full organizational hierarchy and collection of groups, these may not always form the best representation of how teams work. Through an analysis of work, you can get a list of the most closely related people (virtual teams) and files with which a user might be connected. In addition, common utilities, such as those for finding an available meeting time among a set of users, are made available as methods.

Azure Functions

Microsoft Graph exists to be used and customized in broader systems and processes. As a simple REST API and coupled with a wide array of SDKs, Microsoft Graph is designed to be straightforward to work with. A natural choice for building processes and integrations in Microsoft Graph is Azure Functions (functions.azure.com), which lets you add pinpointed blocks of code where you need it while only paying incrementally for code as it's used. Azure Functions supports development across languages, including C# and Node.js.

Recently, a new set of integrations with Azure Functions makes it easier to connect to Microsoft Graph. Azure Functions Binding Extensions, now available in preview with the Azure Functions 2.0 runtime, automates some of the common tasks of working with Microsoft Graph, including authentication and working with the mechanics of webhooks.

Let's take a look at an example to get started in working with Microsoft Graph.

Creating Tasks via Azure Functions

Imagine you'd like to have managers review and approve an action undertaken by a member of their team. User tasks are one way to ask users to perform an action—to convert and track human action. In this case, I want to implement a simple Web service that will create a task assigned to a user's manager.

The first stop in any Microsoft Graph project is usually the Graph Explorer. Graph Explorer is an application Web site that lets you quickly model Microsoft Graph calls, explore their results, and fully conceive of all you might do. Available from developer.microsoft.com/graph, the Graph Explorer lets you either use a read-only demo tenancy, or sign into your own tenancy. You can sign in with your organization account and directly access your own data. We recommend using a developer tenancy, available from the Office Developer Program at dev.office.com/devprogram. This will give you a separate tenancy where you can feel free to experiment with your development.

In this case, you can enter two simple URLs to see the kind of calls you'll be making in this sample. First, you want to check "get a user's manager," which you can see in Graph Explorer by selecting the "GET my manager" sample, shown in **Figure 2**. The URL behind this is shown in the Run Query field.

The second part of the operation is to create a Planner Task. Within Graph Explorer, you can expand the set of samples to add samples of Planner tasks. Within this sample set, you can see the operation for creating a Planner Task (a POST to https://graph.microsoft.com/v1.0/planner/tasks).

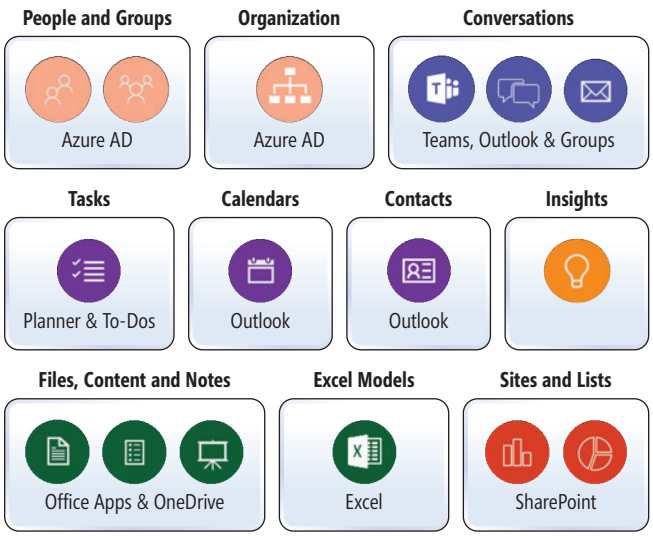
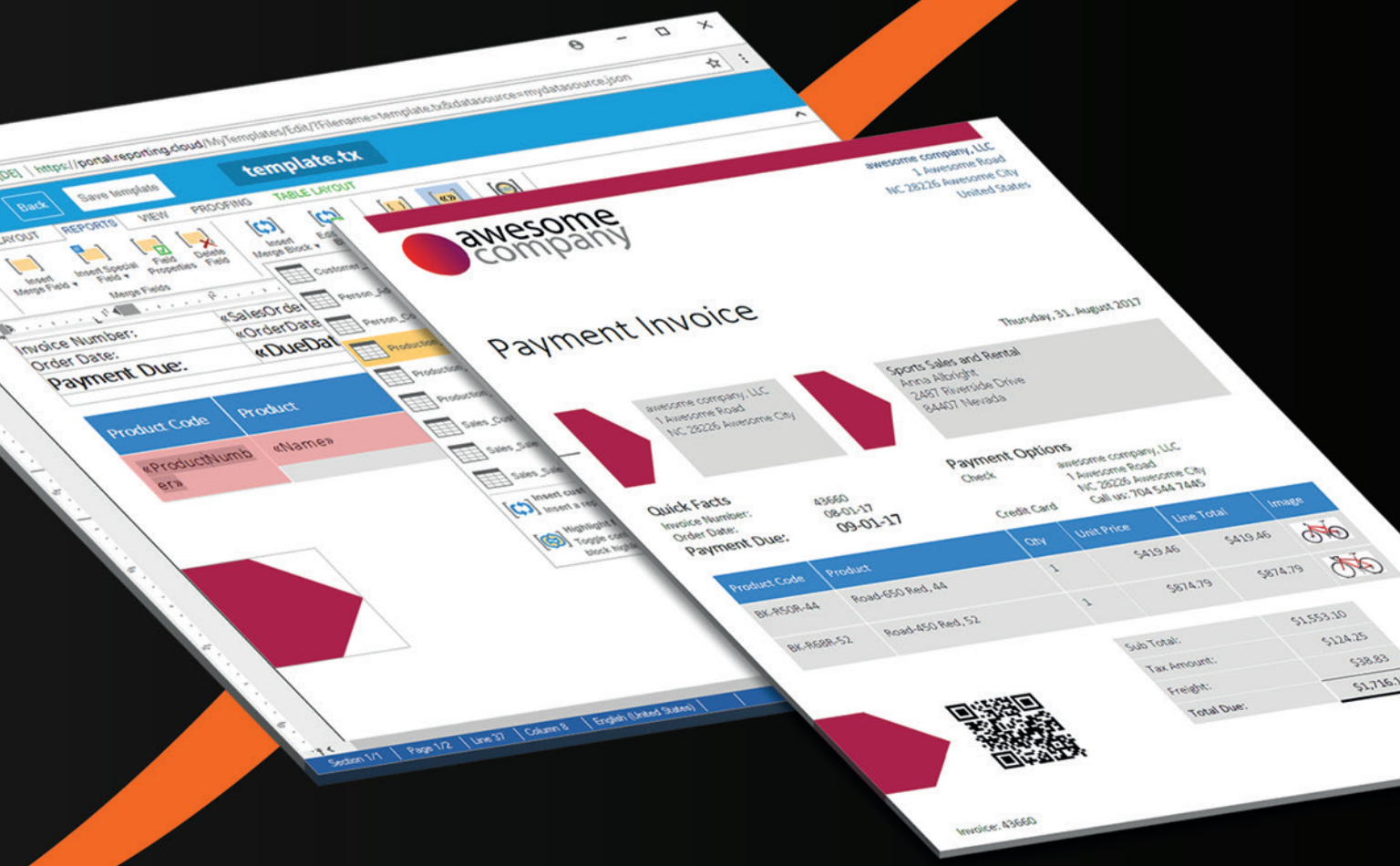


Figure 1 **Productivity APIs of Microsoft Graph**

TEXT CONTROL

GING
OOK AT



Visual Studio
Partner

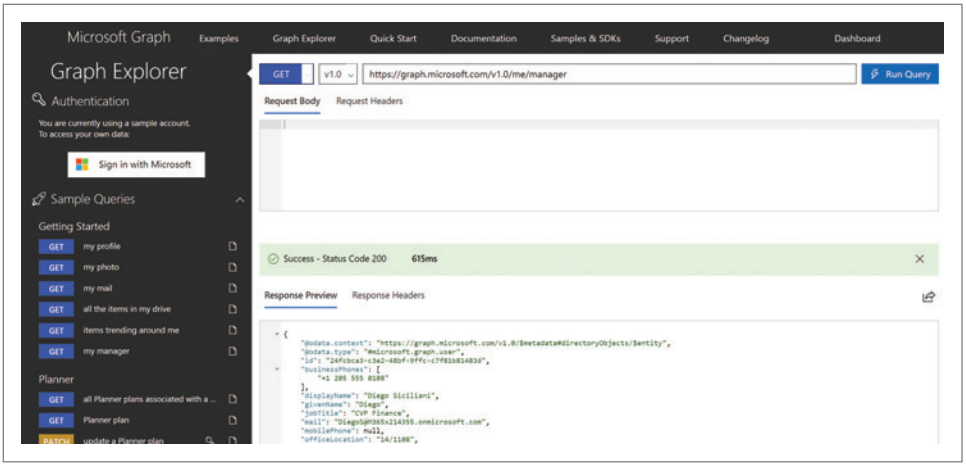


Figure 2 **Results of Selecting GET my manager**

Now that you understand the Web service requests involved, you can build a function using Azure Functions.

To get started, create a new Azure Functions application. In general, you'll want to follow the instructions at aka.ms/azfnmsgraph to get this accomplished. In brief, because the new Azure Functions Binding Extensions capability is in Preview, you'll need to switch your Azure Functions application over to the 2.0 preview ("beta") runtime. You'll also need to install the Microsoft Graph Extension, as well as configure App Service Authentication.

As you configure the Microsoft Graph Application Registration, for this sample you'll need to add some further permissions to support reading manager information and tasks, including:

- CRUD user tasks and projects (Tasks.ReadWrite)
- View users' basic profile (profile)
- Read and Write All Groups (Group.ReadWrite.All)
- Read all users' basic profile (User.ReadBasic.All)

You'll want to leverage Azure Functions Binding Extensions for Microsoft Graph to handle the authentication and ensure you have an authenticated access token by which you can access Microsoft Graph APIs. To do this, you'll create a standard HTTP C# trigger. Under Integrate, select the Advanced Editor and use the bindings shown in **Figure 3**. This will require that the user sign in, authenticate and approve your application before it can be used.

Figure 3 **Creating an HTTP Trigger to Handle Authentication**

```
{
  "bindings": [
    {
      "name": "req",
      "type": "httpTrigger",
      "direction": "in"
    },
    {
      "type": "token",
      "direction": "in",
      "name": "accessToken",
      "resource": "https://graph.microsoft.com",
      "identity": "userFromRequest"
    },
    {
      "name": "$return",
      "type": "http",
      "direction": "out"
    }
  ],
  "disabled": false
}
```

The code for the function is shown in **Figure 4**. Note that you'll need to configure an environment variable for the function application called PlanId, which has the identifier of the Planner Plan you wish to use for your tasks. This can be done via Application Settings for the function application.

This sample shows how you can pull together disparate sets of data (a user's manager and Planner tasks in this case) in one piece of code with one authentication token. Creating and assigning tasks is a common way to drive activities across teams, so the ability to create tasks on the fly and leverage existing Planner experiences is quite useful. It's not quite "widgetMarketingTeam.launchCampaign()"—but at least you can see how you'd create the starter set of tasks that would get the team off to a focused, structured start.

Processing Files in OneDrive

Another task you can perform is to process files that exist on a user's OneDrive. In this instance, you take advantage of Azure Functions Binding Extensions for Microsoft Graph to do the work of preparing a file for use. You then pass it into Cognitive Services APIs for doing voice recognition. This is an example of data processing that can be a useful way to get more value out of files across OneDrive and SharePoint.

To get started, you follow some of the same steps as in the previous example, including setting up a Function App and an Azure Active Directory registration. Note that the Azure Active Directory application registration that you use for this sample will need to have the "Read all files that user can access" (Files.Read.All) permission. You'll also need to have a Cognitive Services Speech API key, which you can obtain from aka.ms/tryspeechapi.

As before, start with Azure Functions Binding Extensions and set up a new HTTP C# trigger. Under the Integrate tab of your function, use the binding markup shown in **Figure 5** to connect your function to a binding extension. In this case, the binding extension ties the myOneDriveFile parameter in your Azure function to the onedrive binding extension.

Now, it's time for the code, which is shown in **Figure 6**.

With this function in place, after a user has signed into their Azure function, they can specify a filename parameter. If a file has a .WAV filename and contains English content within it, this will get transcribed into English text. Because this is implemented with Azure Functions, your function will typically incur cost only as it's called, providing a flexible way to extend the data you have in Microsoft Graph.

Azure Functions + Microsoft Graph

The two samples I presented here show how you can build both human and technical processes on top of data within Microsoft

DocuVieware 3

HTML5 Viewer & Document Management Kit

NEW RELEASE

Easy integration

Full support for custom snap-in

Zero-footprint solution

Fully customizable UI

Mobile devices optimization

Fast & crystal-clear rendering

Check the **New Features** and the **Online Demos**

DOWNLOAD
YOUR FREE TRIAL

www.docuvieware.com

Powered by GdPicture.NET 14

DocuVieware is an ORPALIS imaging technologies product

Figure 4 **Posting a Task Assigned to a User's Manager Azure Functions Source**

```
#r "Newtonsoft.Json"

using System.Net;
using System.Threading.Tasks;
using System.Configuration;
using System.Net.Mail;
using System.IO;
using System.Web;
using System.Text;
using Newtonsoft.Json.Linq;

public static HttpResponseMessage Run(HttpRequestMessage req, string
accessToken, TraceWriter log)
{
  log.Info("Processing incoming task creation requests.");

  // Retrieve data from query string
  // Expected format is taskTitle=task text&taskBucket=bucket
  // title&taskPriority=alert
  var values = HttpUtility.ParseQueryString(req.RequestUri.Query);

  string taskTitle = values["taskTitle"];
  string taskBucket = values["taskBucket"];
  string taskPriority = values["taskPriority"];

  if (String.IsNullOrEmpty(taskTitle))
  {
    log.Info("Incomplete request received - no title.");
    return new HttpResponseMessage(HttpStatusCode.BadRequest);
  }

  string planId = System.Environment.GetEnvironmentVariable("PlanId");

  // Retrieve the incoming users' managers ID
  string managerJson = GetJson(
    "https://graph.microsoft.com/v1.0/me/manager/", accessToken, log);
    dynamic manager = JObject.Parse(managerJson);
  string managerId = manager.id;

  string appliedCategories = "{}";

  if (taskPriority == "alert" || taskPriority == "1")
  {
    appliedCategories = "{ \"category1\": true }";
  }
  else
  {
    appliedCategories = "{ \"category2\": true }";
  }

  string now =  DateTime.UtcNow.ToString("yyyy-MM-ddTHH\\:mm\\:ss.fffffffzzz");
  string due =  DateTime.UtcNow.AddDays(5).ToString(
    "yyyy-MM-ddTHH\\:mm\\:ss.fffffffzzz");
  string bucketId = "";

  // If the incoming request wants to place a task in a bucket,
  // find the bucket ID to add it to
  if (!String.IsNullOrEmpty(taskBucket))
  {
    // Retrieve a list of planner buckets so that you can match
    // the task to a bucket, where possible
    string bucketsJson = GetJson(
      "https://graph.microsoft.com/v1.0/planner/plans/" + planId +
      "/buckets", accessToken, log);

    if (!String.IsNullOrEmpty(bucketsJson))
    {
      dynamic existingBuckets = JObject.Parse(bucketsJson);

      taskBucket = taskBucket.ToLower();

      foreach (var bucket in existingBuckets.value)
      {
        var existingBucketTitle = bucket.name.ToString().ToLower();

        if (taskBucket.IndexOf(existingBucketTitle) >= 0)
        {
          bucketId = ", \"bucketId\": \"" + bucket.id.ToString() + "\"";
        }
      }
    }

    string jsonOutput = String.Format(" {{ \"planId\": \"{0}\",
\"title\": \"{1}\", \"orderHint\": \" !\", \"startDateTime\": \"{2}\",
\"dueDateTime\": \"{6}\", \"appliedCategories\": {3}, \"assignments\":
{{ \"{4}\": {{ \"@odata.type\": \"#microsoft.graph.plannerAssignment\",
\"orderHint\": \" !\"  }} }}{5} }}",
      planId, taskTitle, now, appliedCategories, managerId, bucketId, due);

    log.Info("Creating new task: " + jsonOutput);
    PostJson("https://graph.microsoft.com/v1.0/planner/tasks",
      jsonOutput, accessToken, log);

    return new HttpResponseMessage(HttpStatusCode.OK);
}

private static string GetJson(string url, string token, TraceWriter log)
{
  HttpWebRequest hwr = (HttpWebRequest)WebRequest.CreateHttp(url);

  log.Info("Getting Json from endpoint '" + url + "'");

  hwr.Headers.Add("Authorization", "Bearer " + token);
  hwr.ContentType = "application/json";

  WebResponse response = null;

  try
  {
    response = hwr.GetResponse();
    using (Stream stream = response.GetResponseStream())
    {
      using (StreamReader sr = new StreamReader(stream))
      {
        return sr.ReadToEnd();
      }
    }
  }
  catch (Exception e)
  {
    log.Info("Error: " + e.Message);
  }

  return null;
}

private static string PostJson(string url, string body, string token, TraceWriter log)
{
  HttpWebRequest hwr = (HttpWebRequest)WebRequest.CreateHttp(url);

  log.Info("Posting to endpoint " + url);
  hwr.Method = "POST";
  hwr.Headers.Add("Authorization", "Bearer " + token);
  hwr.ContentType = "application/json";

  var postData = Encoding.UTF8.GetBytes(body.ToString());

  using (var stream = hwr.GetRequestStream())
  {
  stream.Write(postData, 0, postData.Length);
  }

  WebResponse response = null;

  try
  {
    response = hwr.GetResponse();
    using (Stream stream = response.GetResponseStream())
    {
      using (StreamReader sr = new StreamReader(stream))
      {
        return sr.ReadToEnd();
      }
    }
  }
  catch (Exception e)
  {
    log.Info("Error: " + e.Message);
  }

  return null;
}
```

Figure 5 **Setting Up a New Trigger for Getting a File on OneDrive**

```
{
  "bindings": [
    {
      "name": "req",
      "type": "httpTrigger",
      "direction": "in"
    },
    {
      "name": "myOneDriveFile",
      "type": "onedrive",
      "direction": "in",
      "path": "{query.filename}",
      "identity": "userFromRequest",
    },
    {
      "name": "$return",
      "type": "http",
      "direction": "out"
    }
  ],
  "disabled": false
}
```

Graph. Combined with the breadth of coverage of Microsoft Graph and the ability to cross workloads (for example, organizational hierarchy and tasks, as was the case with the task sample in this article), you can build and add value across your entire organization. Combining Microsoft Graph and Azure Functions allows you to build out the full API to your organization, and transform productivity for all. Get started in building solutions for your organization by visiting developer.microsoft.com/graph, and working with Azure Functions at functions.azure.com. ■

Mike Ammerlaan *is a director of product marketing on the Microsoft Office Ecosystem team, helping people build engaging solutions with Office 365. Prior to this, he worked at Microsoft as a program manager for 18 years, developing products such as SharePoint, Excel, Yammer, Bing Maps and Combat Flight Simulator.*

Thanks to the following Microsoft technical experts for reviewing this article: *Ryan Gregg, Matthew Henderson and Dan Silver*

Figure 6 **Transcribing an Audio File from One Drive**

```
#r "Newtonsoft.Json"

using System.Net;
using System.Text;
using System.Configuration;
using Newtonsoft.Json.Linq;

public static  async Task<HttpResponseMessage> Run(HttpRequestMessage req,
  Stream myOneDriveFile, TraceWriter log)
{
  // Download the contents of the audio file
  log.Info("Downloading audio file contents...");
  byte[] audioBytes;

  audioBytes = StreamToBytes(myOneDriveFile);

  // Transcribe the file using cognitive services APIs
  log.Info($"Retrieving the cognitive services access token...");
  var accessToken =
    System.Environment.GetEnvironmentVariable("SpeechApiKey");

  var bingAuthToken = await FetchCognitiveAccessTokenAsync(accessToken);

  log.Info($"Transcribing the file...");
  var transcriptionValue = await RequestTranscriptionAsync(
    audioBytes, "en-us", bingAuthToken, log);

  HttpResponseMessage hrm = new HttpResponseMessage(HttpStatusCode.OK);

  if (null != transcriptionValue)
  {
    hrm.Content = new StringContent(transcriptionValue, Encoding.UTF8, "text/html");
  }
  else
  {
    hrm.Content = new StringContent("Content could not be transcribed.");
  }

  return hrm;
}

private static async Task<string> RequestTranscriptionAsync(byte[] audioBytes,
  string languageCode, string authToken, TraceWriter log)
{
  string conversation_url = $"https://speech.platform.bing.com/speech/
recognition/conversation/cognitiveservices/v1?language={languageCode}";
  string dictation_url = $"https://speech.platform.bing.com/speech/
recognition/dictation/cognitiveservices/v1?language={languageCode}";

  HttpResponseMessage response = null;
  string responseJson = "default";

  try
  {
    response = await PostAudioRequestAsync(conversation_url, audioBytes, authToken);
    responseJson = await response.Content.ReadAsStringAsync();
    JObject data = JObject.Parse(responseJson);
    return data["DisplayText"].ToString();
  }
  catch (Exception ex)
  {
    log.Error($"Unexpected response from transcription service A: {ex.Message} |" +
      responseJson + "|" + response.StatusCode  + "|" +
      response.Headers.ToString() +"|");
    return null;
  }
}

private static async Task<HttpResponseMessage> PostAudioRequestAsync(
  string url, byte[] bodyContents, string authToken)
{
  var payload = new ByteArrayContent(bodyContents);
  HttpResponseMessage response;

  using (var client = new HttpClient())
  {
    client.DefaultRequestHeaders.Add("Authorization", "Bearer " + authToken);
    payload.Headers.TryAddWithoutValidation("content-type", "audio/wav");
    response = await client.PostAsync(url, payload);
  }

  return response;
}

private static byte[] StreamToBytes(Stream stream)
{
  using (MemoryStream ms = new MemoryStream())
  {
    stream.CopyTo(ms);
    return ms.ToArray();
  }
}

private static async Task<string> FetchCognitiveAccessTokenAsync(
  string subscriptionKey)
{
  string fetchUri = "https://api.cognitive.microsoft.com/sts/v1.0";

  using (var client = new HttpClient())
  {
    client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key", subscriptionKey);
    UriBuilder uriBuilder = new UriBuilder(fetchUri);
    uriBuilder.Path += "/issueToken";

    var response = await client.PostAsync(uriBuilder.Uri.AbsoluteUri, null);
    return await response.Content.ReadAsStringAsync();
  }
}
```

Respect the Past. Code the Future.

Visual Studio Live! (VSLive!™) Las Vegas, returns to the strip, March 11 – 16, 2018. During this intense week of developer training, you can sharpen your skills in everything from ASP.NET to Xamarin.

Plus, celebrate 25 years of coding innovation as we take a fun look back at technology and training since 1993. Experience the education, knowledge-share and networking at #VSLive25.

Visual Studio LIVE! 25 YEARS OF CODING INNOVATION

VSLive! 1998

VSLive! 2017

SUPPORTED BY

Microsoft | Visual Studio | msdn magazine | Visual Studio MAGAZINE

PRODUCED BY

1105 MEDIA
YOUR GROWTH. OUR BUSINESS.

DEVELOPMENT TOPICS INCLUDE:

- VS2017/.NET
- Angular/JavaScript
- ASP.NET Core
- Xamarin
- Azure / Cloud
- Hands-On Labs
- Software Practices
- ALM / DevOps
- SQL Server 2017
- UWP (Windows)

Who Should Attend and Why

We've been around since 1993. What's our secret? YOU! Since our first conference (VBITS/VSLive!/Visual Studio Live!), tens of thousands of developers, software architects, programmers, engineers, designers and more have trusted us year-in-and-year-out for unbiased and cutting-edge education on the Microsoft Platform.

Register to code with us today!

Register by January 19 and Save $400!

Use Promo Code VSLJAN4

CONNECT WITH US

twitter.com/vslive – @VSLive

facebook.com – Search "VSLive"

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

vslive.com/lasvegasmsdn

Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS

AGENDA AT-A-GLANCE

BACK BY POPULAR DEMAND
Sunday Pre-Con Hands-On Labs
Choose From:

HOL01 Special 2-Day Hands-On Lab: Modern Security Architecture for ASP.NET Core
Sunday, March 11, 9:00am – 6:00pm (Part 1)*

Monday, March 12, 9:00am – 6:00pm (Part 2)*

Brock Allen
You will learn:

- The security architecture of ASP.NET Core
- About authenticating users with OpenID Connect
- How to protect Web APIs with OAuth2

*This 2-day Hands-On Lab is available with the six-day conference package or on its own. Details at **vslive.com/lasvegasmsdn**.

HOL02 From 0-60 in a Day with Xamarin and Xamarin.Forms
Introductory / Intermediate

Sunday, March 11, 9:00am – 6:00pm

Roy Cornelissen & Marcel de Vries
You will learn:

- How to build your first mobile apps on three platforms with the Xamarin framework
- How to maintain platform uniqueness while sharing a large chunk of your codebase
- How to think "mobile first" in your application architecture

HOL03 Busy Developer's HOL on Angular
Sunday, March 11, 9:00am – 6:00pm

Ted Neward
In this Hands-On Lab, we'll start from zero, with a little TypeScript, then start working with Angular 2: its core constructs and how it works with components, modules, and of course the ubiquitous model/view/controller approach.

ONLY $645 through January 19
Applies to HOL02 and HOL03 only.

ALM / DevOps | Cloud Computing | Database and Analytics | Native Client

START TIME	END TIME	Full Day Hands-On Labs: Sunday, March 11, 2018 *(Separate entry fee required)*
8:00 AM	9:00 AM	Pre-Conference Hands-On Lab Registration - Coffee and Morning Pastries
9:00 AM	6:00 PM	**HOL01** Full Day Hands-On Lab: Modern Security Architecture for ASP.NET Core (Part 1) - *Brock Allen*
4:00 PM	6:00 PM	Conference Registration Open
START TIME	**END TIME**	**Pre-Conference Workshops: Monday, March 12, 2018** *(Separate entry fee required)*
7:30 AM	9:00 AM	Pre-Conference Workshop Registration - Coffee and Morning Pastries
9:00 AM	6:00 PM	**HOL01** Full Day Hands-On Lab: Modern Security Architecture for ASP.NET Core (Part 2) - *Brock Allen*
7:00 PM	9:00 PM	Dine-A-Round
START TIME	**END TIME**	**Day 1: Tuesday, March 13, 2018**
7:00 AM	8:00 AM	Registration - Coffee and Morning Pastries
8:00 AM	9:15 AM	**T01** Go Serverless with Azure Functions - *Eric D. Boyd* / **T02** Getting Ready to Write Mobile Applications with Xamarin - *Kevin Ford*
9:30 AM	10:45 AM	**T05** Angular 101 - *Deborah Kurata* / **T06** Lessons Learned from Real World Xamarin.Forms Projects - *Nick Landry*
11:00 AM	12:00 PM	**KEYNOTE:** .NET Everywhere and for Everyone - *James Montemagno, Principal*
12:00 PM	1:00 PM	Lunch
1:00 PM	1:30 PM	Dessert Break - Visit Exhibitors
1:30 PM	2:45 PM	**T09** Busy Developer's Guide to Chrome Development - *Ted Neward* / **T10** Works On My Machine... Docker for Developers - *Chris Klug*
3:00 PM	4:15 PM	**T13** Angular Component Communication - *Deborah Kurata* / **T14** Customizing Your UI for Mobile Devices: Techniques to Create a Great User Experience - *Laurent Bugnion*
4:15 PM	5:30 PM	Welcome Reception
START TIME	**END TIME**	**Day 2: Wednesday, March 14, 2018**
7:30 AM	8:00 AM	Registration - Coffee and Morning Pastries
8:00 AM	9:15 AM	**W01** The Whirlwind Tour of Authentication and Authorization with ASP.NET Core - *Chris Klug* / **W02** Building Mixed Reality Experiences for HoloLens & Immersive Headsets in Unity - *Nick Landry*
9:30 AM	10:45 AM	**W05** TypeScript: The Future of Front End Web Development - *Ben Hoelting* / **W06** A Dozen Ways to Mess Up Your Transition From Windows Forms to XAML - *Billy Hollis*
11:00 AM	12:00 PM	**General Session:** To Be Announced - *Kasey Uhlenhuth, Program Manager, .NET &*
12:00 PM	1:00 PM	Birds-of-a-Feather Lunch
1:00 PM	1:30 PM	Dessert Break - Visit Exhibitors - Exhibitor Raffle @ 1:15pm (Must be present to win)
1:30 PM	1:50 PM	**W09** Fast Focus: 0-60 for Small Projects in Visual Studio Team Services - *Alex Mullans*
2:00 PM	2:20 PM	**W12** Fast Focus: HTTP/2: What You Need to Know - *Robert Boedigheimer*
2:30 PM	3:45 PM	**W15** Advanced Fiddler Techniques - *Robert Boedigheimer* / **W16** Building Cross-Platforms Business Apps with C# and CSLA .NET - *Rockford Lhotka*
4:00 PM	5:15 PM	**W19** Assembling the Web - A Tour of WebAssembly - *Jason Bock* / **W20** Radically Advanced XAML: Dashboards, Timelines, Animation, and More - *Billy Hollis*
7:00 PM	8:30 PM	VSLive! High Roller Evening Out
START TIME	**END TIME**	**Day 3: Thursday, March 15, 2018**
7:30 AM	8:00 AM	Registration - Coffee and Morning Pastries
8:00 AM	9:15 AM	**TH01** ASP.NET Core 2 For Mere Mortals - *Philip Japikse* / **TH02** Performance in 60 Seconds – SQL Tricks Everybody MUST Know - *Pinal Dave*
9:30 AM	10:45 AM	**TH05** Getting to the Core of ASP.NET Core Security - *Adam Tuliper* / **TH06** Secrets of SQL Server - Database Worst Practices - *Pinal Dave*
11:00 AM	12:00 PM	Panel Discussion: To Be Announced
12:00 PM	1:00 PM	Lunch
1:00 PM	2:15 PM	**TH09** Entity Framework Core 2 For Mere Mortals - *Philip Japikse* / **TH10** SQL Server 2017 - Intelligence Built-in - *Scott Klein*
2:30 PM	3:45 PM	**TH13** MVVM and ASP.NET Core Razor Pages - *Ben Hoelting* / **TH14** Introduction to Azure Machine Learning - *James McCaffrey*
4:00 PM	5:15 PM	**TH17** Securing Web Apps and APIs with IdentityServer - *Brian Noyes* / **TH18** Introduction to the CNTK v2 Machine Learning Library - *James McCaffrey*
START TIME	**END TIME**	**Post-Conference Workshops: Friday, March 16, 2018** *(Separate entry fee required)*
7:30 AM	8:00 AM	Post-Conference Workshop Registration - Coffee and Morning Pastries
8:00 AM	5:00 PM	**F01** Workshop: Creating Mixed Reality Experiences for HoloLens & Immersive Headsets with Unity - *Nick Landry & Adam Tuliper*

Speakers and sessions subject to change

Visual Studio LIVE! 25 YEARS OF CODING INNOVATION

March 11 – 16, 2018
Bally's Hotel & Casino

LasVegas

Software Practices	Visual Studio / .NET Framework	Web Client	Web Server

Full Day Hands-On Labs: Sunday, March 11, 2018 *(Separate entry fee required)*

Pre-Conference Hands-On Lab Registration - Coffee and Morning Pastries

HOL02 Full Day Hands-On Lab: From 0-60 in a Day with Xamarin and Xamarin.Forms - *Roy Cornelissen & Marcel de Vries*	**HOL03** Full Day Hands-On Lab: Busy Developer's HOL on Angular - *Ted Neward*

Conference Registration Open

Pre-Conference Workshops: Monday, March 12, 2018 *(Separate entry fee required)*

Pre-Conference Workshop Registration - Coffee and Morning Pastries

M02 Workshop: Developer Dive into SQL Server 2016 - *Leonard Lobel*	**M03** Workshop: Add Intelligence to Your Solutions with AI, Bots, and More - *Brian Randell*

Dine-A-Round

Day 1: Tuesday, March 13, 2018

Registration - Coffee and Morning Pastries

T03 Database Development with SQL Server Data Tools - *Leonard Lobel*	**T04** What's New in Visual Studio 2017 for C# Developers - *Kasey Uhlenhuth*
T07 Introduction to Azure Cosmos DB - *Leonard Lobel*	**T08** Using Visual Studio Mobile Center to Accelerate Mobile Development - *Kevin Ford*

Program Manager – Xamarin, Microsoft

Lunch

Dessert Break - Visit Exhibitors

T11 DevOps for the SQL Server Database - *Brian Randell*	**T12** To Be Announced
T15 PowerShell for Developers - *Brian Randell*	**T16** To Be Announced

Welcome Reception

Day 2: Wednesday, March 14, 2018

Registration - Coffee and Morning Pastries

W03 Using Feature Toggles to Separate Releases from Deployments - *Marcel de Vries*	**W04** Lock the Doors, Secure the Valuables, and Set the Alarm - *Eric D. Boyd*
W07 Overcoming the Challenges of Mobile Development in the Enterprise - *Roy Cornelissen*	**W08** Computer, Make It So! - *Veronika Kolesnikova & Willy Ci*

Visual Studio, Microsoft

Birds-of-a-Feather Lunch

Dessert Break - Visit Exhibitors - Exhibitor Raffle @ 1:15pm (Must be present to win)

W10 Fast Focus: Cross Platform Device Testing with xUnit - *Oren Novotny*	**W11** Fast Focus: Understanding .NET Standard - *Jason Bock*
W13 Fast Focus: Serverless Computing: Azure Functions and Xamarin in 20 minutes - *Laurent Bugnion*	**W14** Fast Focus: TBD - *Scott Klein*
W17 Versioning NuGet and npm Packages - *Alex Mullans*	**W18** Getting to the Core of .NET Core - *Adam Tuliper*
W21 Encrypting the Web - *Robert Boedigheimer*	**W22** Porting MVVM Light to .NET Standard: Lessons Learned - *Laurent Bugnion*

VSLive! High Roller Evening Out

Day 3: Thursday, March 15, 2018

Registration - Coffee and Morning Pastries

TH03 Demystifying Microservice Architecture - *Miguel Castro*	**TH04** Cognitive Services in Xamarin Applications - *Veronika Kolesnikova*
TH07 Unit Testing Makes Me Faster: Convincing Your Boss, Your Co-Workers, and Yourself - *Jeremy Clark*	**TH08** Publish Your Angular App to Azure App Services - *Brian Noyes*

Panel Discussion: To Be Announced

Lunch

TH11 Writing Testable Code and Resolving Dependencies - DI Kills Two Birds with One Stone - *Miguel Castro*	**TH12** Signing Your Code the Easy Way - *Oren Novotny*
TH15 "Doing DevOps" as a Politically Powerless Developer - *Damian Brady*	**TH16** Analyzing Code in .NET - *Jason Bock*
TH19 I'll Get Back to You: Task, Await, and Asynchronous Methods - *Jeremy Clark*	**TH20** Multi-targeting the World: A Single Project to Rule Them All - *Oren Novotny*

Post-Conference Workshops: Friday, March 16, 2018 *(Separate entry fee required)*

Post-Conference Workshop Registration - Coffee and Morning Pastries

F02 Workshop: Distributed Cross-Platform Application Architecture - *Jason Bock & Rockford Lhotka*	**F03** Workshop: UX Design for Developers: Basics of Principles and Process - *Billy Hollis*

Bally's Hotel & Casino will play host to Visual Studio Live!, and is offering a special reduced room rate to conference attendees.

CONNECT WITH VISUAL STUDIO LIVE!

- twitter.com/vslive – @VSLive
- facebook.com – Search "VSLive"
- linkedin.com – Join the "Visual Studio Live" group!

REGISTER NOW

vslive.com/lasvegasmsdn

What's New for .NET UWP Development?

Daniel Jacobson and Stefan Wick

Back in October 2015, one of us had the pleasure to author an article about .NET Universal Windows Platform (UWP) development (msdn.com/magazine/mt590967). A lot has changed with .NET UWP development since then, and we wanted to address those changes with a new article that helps get developers up to speed.

Since October 2015, Microsoft has published a new release of Visual Studio 2017, several new SDKs (including the most recent Windows 10 Fall Creators Update SDK), the Windows Application Packaging Project, .NET Standard 2.0 support, improvements to NuGet, the new Fluent design system, new community driven tools including Windows Template Studio, and more. Many of these changes and improvements to the UWP developer ecosystem are designed to make it easier to bring your existing .NET assets forward, optimize your application deployment strategy on Windows 10, and accelerate your UWP development to build great applications as quickly as possible.

The Windows 10 Fall Creators Update brings a host of improvements for UWP developers, with Visual Studio 2017 version 15.5 and later providing the best support for the Windows 10 Fall Creators Update SDK (10.0.16299.0). Some of the most important changes include new capabilities for enterprise applications, streamlined application deployment, and support for both .NET Standard 2.0 and the new Fluent Design System. Let's explore these advancements.

This article discusses:

- Improvements for UWP developers in the Windows 10 Fall Creators Update
- How developers can modernize WPF and Windows Forms applications with Windows 10 and the UWP
- What's new in Visual Studio 2017 for Windows client application developers
- Support for .NET Standard 2.0 in UWP development projects

Technologies discussed:

Visual Studio 2017, NuGet, .NET, Universal Windows Platform, Windows Presentation Foundation, Windows Forms

New Capabilities in Windows for Enterprise Applications

Microsoft recognized the need to improve the Windows 10 Platform for enterprise developers building applications for Windows on desktop PCs and has made important strides in this area with the Windows 10 Fall Creators Update.

Best of Both UWP and Win32 Microsoft has improved the Windows 10 Platform with the Desktop Bridge (aka.ms/desktopbridge) to enhance its appeal to all .NET developers, whether their current focus is on the UWP, Windows Presentation Foundation (WPF), Windows Forms or Xamarin. With the new App Packaging project type in Visual Studio 2017 version 15.5, you can create Windows App Packages for your WPF or Windows Forms projects, just like you can for UWP projects.

Once your application is packaged, you get all the Windows 10 application deployment benefits, including the option to distribute via the Microsoft Store (for consumer apps), the Microsoft

Store for Business and Education, or via any of your favorite app deployment options such as Intune. Packaged apps have access to both the full UWP API surface and the Win32 APIs on desktop, so you can modernize your WPF and Windows Forms applications gradually with UWP APIs and Windows 10 features. You can also include your Win32 components in your UWP apps so that they light up on the desktop with all Win32 capabilities.

The recent Fall Creators Update release has helped close the API and feature gaps between UWP and Win32 desktop applications.

Peace of Mind Users can install apps without regret and without concern that they contain malware. Installations don't require admin privileges, and each application install is isolated from others, easing management. Uninstalls are guaranteed to be clean, with no rot in the Registry or file system or other unexpected side effects from installing software. This method of distribution is now also available for your WPF and Windows Forms applications using the Windows Application Packaging Project.

Enterprise-Ready Security Windows Hello enables biometric authentication for devices and compatible apps. The Windows 10 App Model protects your data and system health through UWP app containers, which put the user in control over what apps can and cannot do. In addition, apps must disclose their usage of resources, such as location or microphone, which the user or IT administrator controls via privacy settings.

Closing Gaps for Desktop Applications The recent Fall Creators Update release has helped close the API and feature gaps between UWP and Win32 desktop applications. Support for .NET Standard 2.0 has significantly increased the available API surface, while providing access to many important NuGet packages that previously could not be used in UWP projects. For example, enterprise developers finally get access to SqlClient APIs to talk directly to SQL Server databases from their UWP projects. Microsoft also continues to improve the UWP App Model for desktop and enterprise scenarios. Recent updates have enabled command-line activation, auto-startup and capabilities for running with full-trust or with unrestricted execution lifetime (extendedBackgroundTaskTime and extendedExecutionUnconstrained).

Streamlined Application Deployment

Microsoft recognizes that many .NET developers are still building Windows Forms and WPF applications. In addition to making it as easy as possible to bring existing assets forward, the company aims to solve a problem that many developers face today: deployment. With Visual Studio 2017, it's easier than ever to leverage the Desktop Bridge (aka.ms/desktopbridge) to package existing Win32 .NET applications.

You can certainly submit Win32 applications to the Microsoft Store with the Windows Application Packaging Project, but there are many other ways to distribute applications packaged as an .appx. These options support the deployment flexibility that many organizations require.

Beginning with the Windows 10 Fall Creators Update, you can host your own .appx installer on the Web to invoke the app installer automatically. It's as simple as adding an activation scheme (ms-appinstaller:?source=) to the link to the .appx on the Web. For more information, check out the blog post on direct Web installs here: bit.ly/2mJflUl. This approach makes it really easy to share applications within your organization, simply by posting a link to the .appx on a Web site, for example.

You can also create app installer manifests to support automatic package updates. One advantage that this app installer technology provides over a scheme like ClickOnce is that automatic updates happen behind the scenes--the application can update before the user of the application even launches it. To learn how to create your own custom .appinstaller files for automatic application updates in privately hosted deployment channels, check out the blog here: bit.ly/2z4Fx3A.

If you need more capable device management tools in your organization, you can leverage and distribute your applications with Microsoft Intune (bit.ly/2B7OnyR). The Windows Application Packaging Project is the fastest route to make your apps easily managed via Microsoft Intune.

Last but not least, you can distribute your applications via the Visual Studio App Center (bit.ly/2AKpsjJ). Visual Studio App Center allows for rapid deployment to your development teams or beta testers of your application. You can also easily enable analytics to determine who's using your applications, and how they're using them.

With Visual Studio 2017, it's easier than ever to leverage the Desktop Bridge to package existing Win32 .NET applications.

With the Windows Application Packaging Project and so many channels of distribution for .appx packages, it's easy to package and distribute your application for any scenario in your organization.

Support for .NET Standard 2.0

The Windows 10 Fall Creators Update is the first release of Windows 10 to provide support for .NET Standard 2.0. If you're unfamiliar with .NET Standard, you can get a lot of detail at aka.ms/dotnetstandard. Effectively, .NET Standard is a reference implementation of the base class library that any .NET platform can implement, be it .NET Framework, .NET Core or Xamarin. The goal of .NET Standard is to make it as easy as possible for .NET developers to share code across any .NET platform on which they choose to work.

While there are some similarities with the Portable Class Library (PCL) model, the biggest difference with .NET Standard is that you don't have to choose which platforms you're targeting. If the

.NET Standard	1.0	1.1	1.2	1.3	1.4	1.5	1.6	2.0
.NET Core	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0
.NET Framework	4.5	4.5	4.5.1	4.6	4.6.1	4.6.1 ~~4.6.2~~	4.6.1 ~~vNext~~	4.6.1
Mono	4.6	4.6	4.6	4.6	4.6	4.6	4.6	5.4
Xamarin.iOS	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.14
Xamarin.Mac	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.8
Xamarin.Android	7.0	7.0	7.0	7.0	7.0	7.0	7.0	8.0
Universal Windows Platform	10.0	10.0	10.0	10.0	10.0	10.0.16299	10.0.16299	10.0.16299
Windows	8.0	8.0	8.1					
Windows Phone	8.1	8.1	8.1					
Windows Phone Silverlight	8.0							

Figure 1 **.NET Standard Compatibility Matrix**

platform you target implements the base class library as defined by the standard, the shared code will work. A huge advantage of .NET Standard compared to PCLs is that when a new .NET Platform is introduced with .NET Standard support, you don't need to modify your shared code libraries, because you won't have to add any new platform target.

Early iterations of .NET Standard (versions 1.x) were a great start, but with the Windows 10 Fall Creators Update and .NET Standard 2.0, Microsoft has made it drastically easier to share code across .NET platforms. .NET Standard 2.0 has about 20,000 more APIs compared to any 1.x version. In addition, .NET Standard 2.0 is supported with the .NET Framework version 4.6.1, as well as UWP development for developers targeting the Windows 10 Fall Creators Update. We'll explore the nitty gritty details regarding how all this works in the ".NET and UWP Development" section.

One other huge benefit of .NET Standard 2.0 is that it provides a path forward for many existing .NET applications (whether they're WPF or Windows Forms). If your .NET apps target .NET Framework 4.6.1 or later, you can start porting your codebehind into .NET Standard 2.0 libraries. That same code can then be shared across any .NET Platform that implements .NET Standard 2.0. The .NET Standard Compatibility Matrix in **Figure 1** shows how support plays out across versions.

This means you can start to port your code into reusable libraries, and when you're ready, you can share the same code in UWP apps, Xamarin apps, .NET Core apps and more.

> One other huge benefit of .NET Standard 2.0 is that it provides a path forward for many existing .NET applications.

To assist you in determining if your code is API-compatible with .NET Standard 2.0, Microsoft created a portability analyzer that will produce a report showing which APIs are compatible with each .NET implementation (bit.ly/2zlgrBz). You can also check out a useful Channel9 video from .NET Conf 2017 that steps through the entire porting process, from a Windows Forms .NET application to a UWP application at bit.ly/2j9TB52.

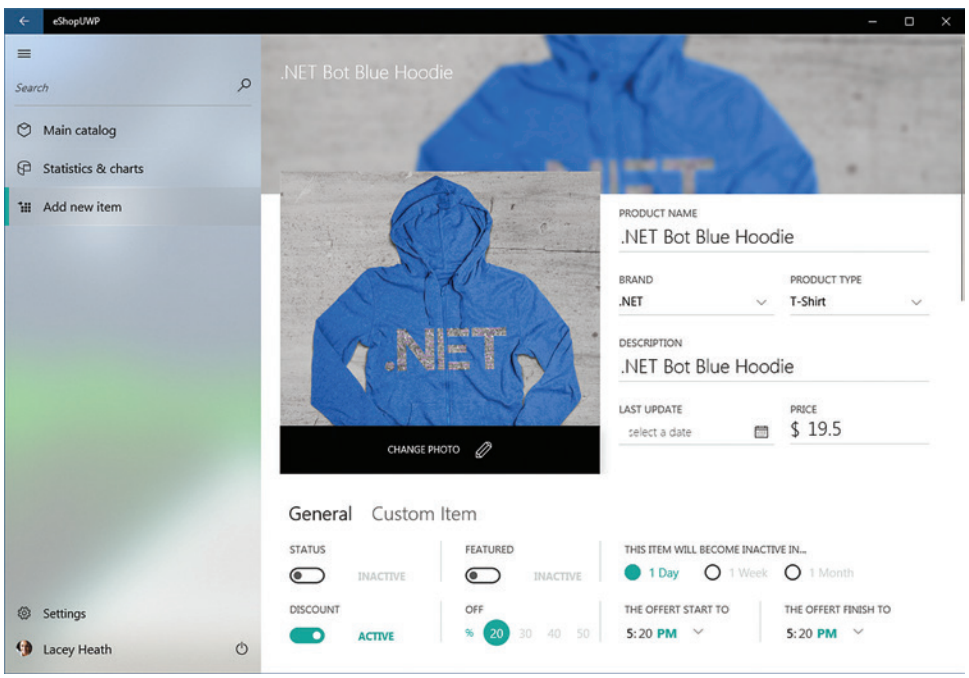


Figure 2 **Fluent Design in Action**

Fluent Design System

With the evolution of computing devices and the emergence of mixed reality, it was essential that the Windows design system evolve, as well. To address this shift, Microsoft released the Fluent design system (fluent.microsoft.com). Built on top of five building block concepts—light, depth, motion, material and scale—this system is designed to support the next generation of UX across device types.

With the Windows 10 Fall Creators Update, you can already see Fluent artifacts making their way into Windows 10 and application

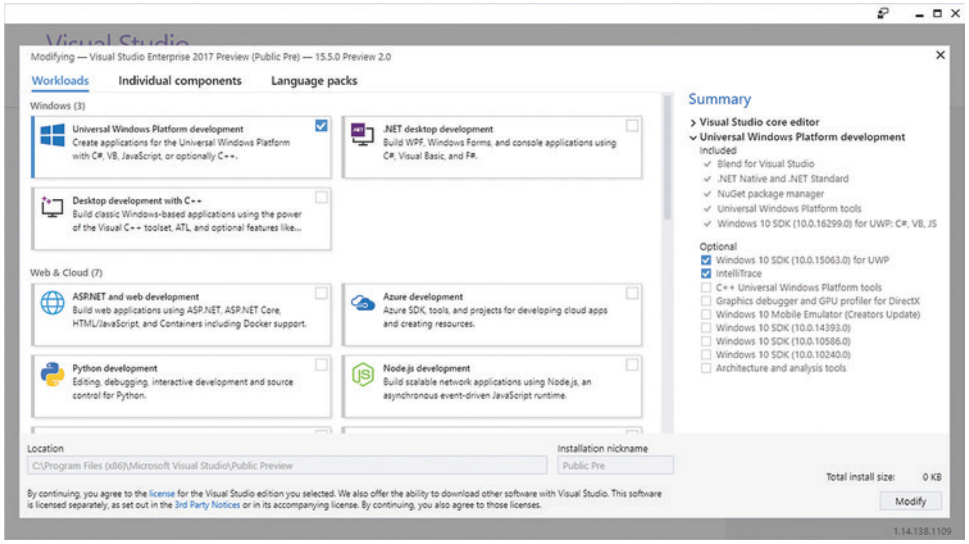


Figure 3 **Visual Studio Installer**

development. A great way to see Fluent design in action is to take a look at the XAMLUIBasics sample in the UWP samples on GitHub (bit.ly/2mK4pvH).

> One of the core priorities for Visual Studio 2017 was to make setup as fast and as easy as possible.

Another sample (shown in **Figure 2**) that demonstrates Fluent design principles in a more enterprise-focused scenario can be found at aka.ms/eshopuwp/src. In this example, you can see the acrylic material that provides a composited layer in the navigation pane. You can also see light effects that serve to highlight the "Add new item" navigation button.

Improvements to Visual Studio 2017

There are many improvements in Visual Studio 2017 compared to Visual Studio 2015. Here, we'll highlight some of the most important and impactful new capabilities of the IDE.

Let's start with the latest tooling—Visual Studio 2017. One of the core priorities for Visual Studio 2017 was to make setup as fast and as easy as possible. As soon as you begin to install Visual Studio 2017, you'll immediately recognize improvements over Visual Studio 2015.

Things get started with the Visual Studio Installer (see **Figure 3**), which allows you to manage all installations of Visual Studio 2017. It supports scenario-driven acquisition with the introduction of Workloads designed around what developers need to get their job done. Workloads make it so that you only install what you need, and not all the extra stuff that often made Visual Studio 2015 take so long to install. In addition, instances of Visual Studio 2017 can be installed side-by-side, so you can install preview versions (visualstudio.com/vs/preview) and retail versions (visualstudio.com) on the same machine. This lets you try out the latest and greatest features without impacting your production environment.

In addition, beginning with the Windows 10 Creators Update (10.0.15063.0), the SDK install can be set up side-by-side, as well. Now you can enroll in the Windows Insiders Program (insider.windows.com) and try out preview SDKs, again without breaking your production environment.

Improvements to the XAML Designer

Visual Studio 2017 version 15.4 introduced some drastic updates to the XAML Designer. These updates are centered on several design considerations, including:

- Significantly improved designer performance
- High-fidelity designer surface, focused especially on rendering artifacts introduced by the new Windows Fluent Design System
- Fewer designer exceptions to minimize distractions and maximize developer productivity in the design surface
- Improved designer UX without breaking or changing existing XAML development in Visual Studio and Blend
- Updated tools that help maximize the productivity of developers building Windows experiences.

These updates to the designer required a rethink of the designer's underlying architecture. The impact on Windows developers in the short term is that Microsoft must rebuild much of the tooling with which developers have grown familiar. To manage this challenge, Microsoft has only released the updates to the XAML Designer to UWP developers targeting the Windows 10 Fall Creators Update (10.0.16299.0) or later. You can read about the updates in much greater detail at aka.ms/uwpsiblog.

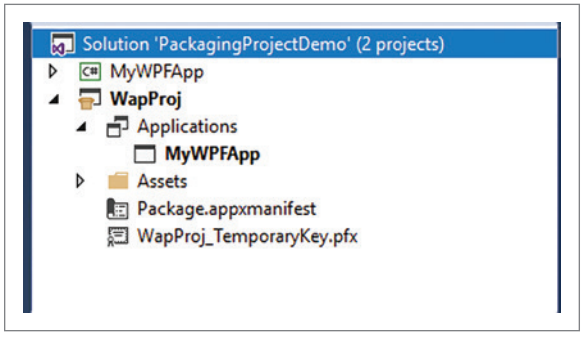
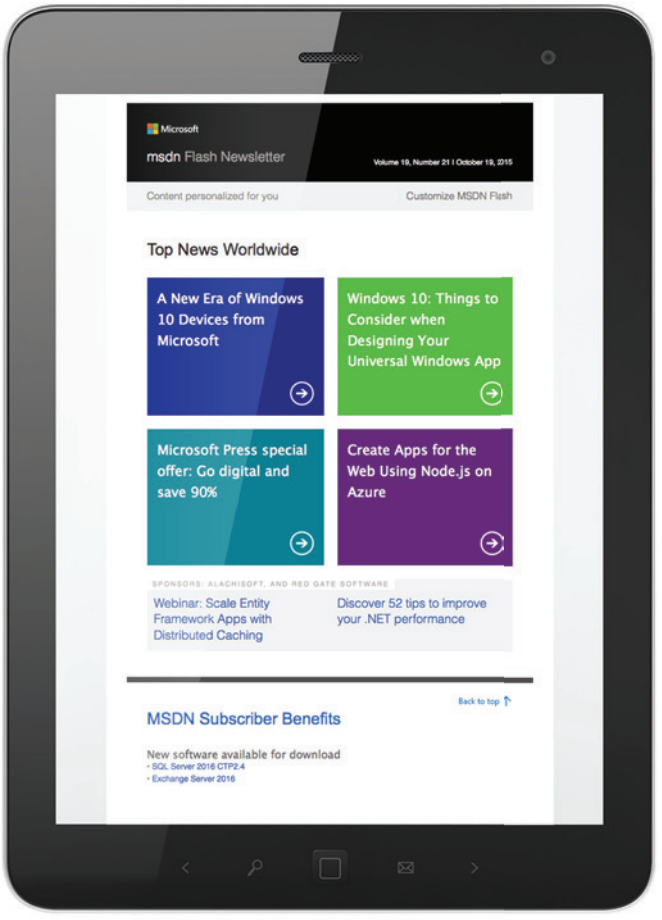


Figure 4 **WapProj Demo**

Windows Application Packaging Project

To streamline application deployment with Windows 10, you can now create a new project type called the Windows Application Packaging



Get news from MSDN in your inbox!

Sign up to receive **MSDN FLASH**, which delivers the latest resources, SDKs, downloads, partner offers, security news, and updates on national and local developer events.

msdn magazine

msdn.microsoft.com/flashnewsletter

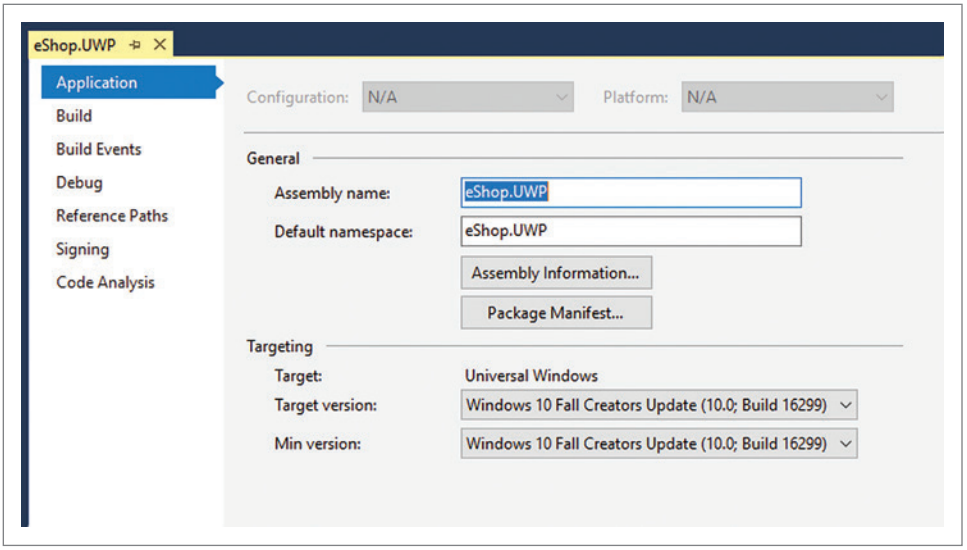


Figure 5 **.NET Standard 2.0-Compatible UWP App**

Project. This project allows you to add references to Win32 applications (for example, Windows Forms and WPF) and package them using the .appx package format.

> MSBuild integration enables conditional package references, which provides finer control over your project dependencies.

Taking advantage of this project is simple. Just add a new Windows Application Packaging Project to your solution and add a reference to your Win32 application using the Applications node, as shown in **Figure 4**.

To create packages, right-click on the Windows Application Packaging Project and select Store | Create App Packages. The output will be an .appx that you can deploy using any of the methods mentioned in the "Streamlined Application Deployment" section of this article.

Improvements to NuGet

With Visual Studio 2017, all .NET UWP apps take advantage of the NuGet package reference model for adding NuGet packages to projects (bit.ly/2AXDDlz). NuGet package references provide several benefits for both NuGet package authors and NuGet package consumers.

PackageReference has direct MSBuild integration, which enables new scenarios such as package management across multiple projects. Let's consider an example. Say you work in an enterprise that has a core library shared across all (or many) projects that your enterprise produces. Traditionally, when an update to that library was made, every project that consumes that package would need to update the reference to take advantage of the latest version. With PackageReference, your enterprise can create a shared MSBuild target that gets imported into every project, and when an update to your core libraries are made, you can update the version in the custom MSBuild target. Every project that imports that MSBuild target will now reference the updated version of the library with no need to update every single project.

In addition, MSBuild integration enables conditional package references, which provides finer control over your project dependencies. Let's consider the same example, but now producing both a Debug and Release version of the core library. The shared MSBuild target could specify conditional package dependencies to consume the Debug version of the library in Debug builds of the projects that consume the library. This way, you have the unoptimized version of the library to provide the best debugging experience. You can configure the conditional dependency to then switch to the Release version of the core library when building Release versions of the project.

Finally, for NuGet package authors, you can leverage platform multi-targeting in a single NuGet package. One great use case of platform multi-targeting is publishing a single NuGet package to

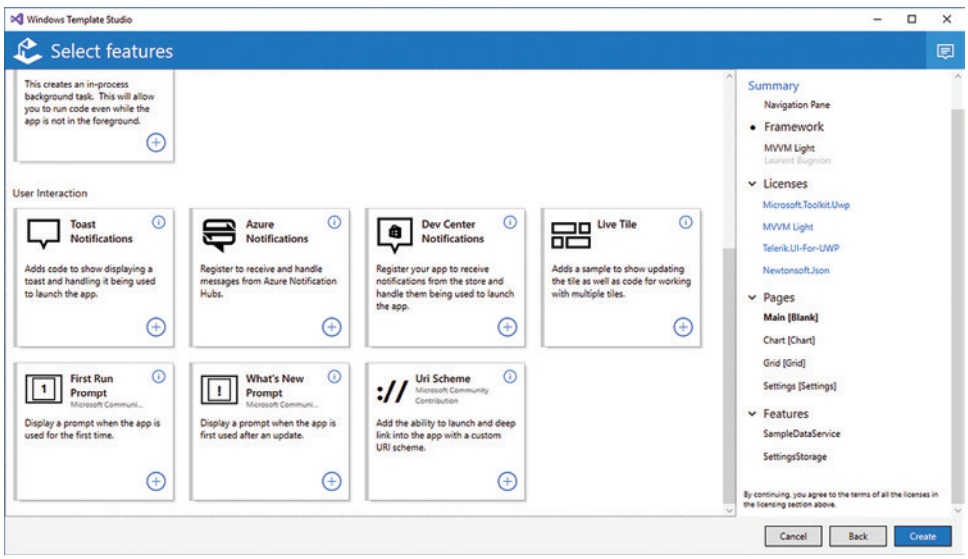


Figure 6 **Windows Template Studio Project Generation**

support all .NET platforms. Let's say you've previously built and managed a .NET Framework library on NuGet that you want to update to support .NET Standard 2.0. The latest version of your NuGet package can maintain both implementations so the latest version of the package is always applicable, regardless of the consuming project. For example, if you have a .NET Framework 4.5 implementation and a .NET Standard 2.0 implementation, any .NET application that's targeting .NET Framework 4.5 or later *or* implements .NET Standard 2.0 will be able to consume the NuGet package. For UWP development, the specified platform is correlated to the TargetPlatformMinVersion of your project. If your TargetPlatformMinVersion is greater than or equal to the platform specified in the NuGet package, the NuGet package can be referenced in the project.

In Visual Studio 2015, the .NET Native compiler was shipped as part of Visual Studio. Even though the .NET Core implementation for UWP development was shipped as a set of NuGet packages, developers faced a slightly fragmented experience when it came to .NET Native. With Visual Studio 2017, all of .NET for UWP development ships as part of the .NET implementation for the UWP NuGet package, including .NET Native. This gives you the flexibility to move at your pace—by specifying the version of .NET that you need and that you know works for your application.

.NET Standard 2.0 and UWP Development

Previously mentioned in this article is that .NET Standard 2.0 is available in the Windows 10 Fall Creators update. To take advantage of the .NET Standard 2.0 implementation, you must set your TargetPlatformMinVersion of your UWP project to the Windows 10 Fall Creators Update or later, as shown in **Figure 5**. You must also reference the Microsoft.NETCore.UniversalWindowsPlatform NuGet package version 6.0 or later.

The Microsoft.NETCore.UniversalWindowsPlatform NuGet package takes advantage of platform multi-targeting support for UWP development so you can always update to the latest version of the package. If you are building an application that has a TargetPlatformMinVersion less than the Windows 10 Fall Creators Update, the NuGet restore process will bring in the .NET Standard 1.4 implementation of .NET Core for Windows 10. If you set the TargetPlatformMinVersion to the Windows 10 Fall Creators Update or later, NuGet will bring in the .NET Standard 2.0 implementation of .NET Core for Windows 10. Now, that was a lot of text—what it means for you is that you shouldn't have to worry about which version of the NuGet package to reference. You can always reference the latest and it should always work, regardless of your Windows 10 target.

Useful Open Source Projects

Finally, we wanted to leave you with some helpful open source projects that you can use to accelerate UWP development. These include Windows Template Studio, the UWP Community Toolkit and the Telerik Controls for UWP Development.

Windows Template Studio Windows Template Studio (shown in **Figure 6**) is a dynamic template generator for UWP development. It's the fastest way to get started building a production-ready UWP app. It leverages best practices for development, and lets you build many views, background tasks, and more in a simple wizard experience. Using Windows Template Studio for all new application development in an organization can help ensure consistent look and feel across apps and encourage best practices for application architectures. Built, maintained and updated by a passionate community of Windows developers, Windows Template Studio is a great way to get started building fully-fledged UWP apps (aka.ms/wts).

With .NET Standard 2.0 and all the free resources available, it's never been easier to build enterprise-ready applications with the UWP.

Telerik Controls for UWP Development Back in February 2017, Telerik announced that it was open sourcing the Telerik UI for UWP controls library. The license provides free use for anyone building UWP apps. Telerik has a history of building powerful, performant, enterprise-ready controls—and with UWP development they're now free! Read more about the Telerik open source announcement on the company's blog: bit.ly/2AZA0il.

UWP Community Toolkit The UWP Community Toolkit (github.com/Microsoft/UWPCommunityToolkit) is another community-driven project that provides helpful tools for UWP development. The toolkit includes templated animations, controls, helper classes, services and more. Use the UWP Community Toolkit to accelerate your development.

Closing Thoughts

It's clear that a lot has changed for .NET and UWP developers, with so many new tools and updates to make building Windows applications as fast and easy as possible. With .NET Standard 2.0 and all the free resources available, it's never been easier to build enterprise-ready applications with the Universal Windows Platform. For .NET developers, the Desktop Bridge platform and updated tooling combine to let you build powerful desktop applications that take advantage of the best of UWP and Win32, including WPF and Windows Forms. Microsoft is constantly improving its tools based on your feedback, so as you build your Windows applications, don't hesitate to reach out by e-mail or Twitter. ■

Daniel Jacobson *is a program manager for Visual Studio, working on tools for Windows platform developers. Reach him at dajaco@microsoft.com or on Twitter: @pmatmic.*

Stefan Wick *is a program manager lead in Windows, working on the Universal Windows Platform. Reach him at swick@microsoft.com or on Twitter: @StefanWickDev.*

Thanks to the following Microsoft technical experts for reviewing this article: *Mike Harsh, Matthijs Hoekstra, Unni Ravindranathan and Ricardo (Rido) Minguez*

AUGUST 6 – 10, 2018 • Microsoft Headquarters, Redmond, WA

Change is ~~Coming~~. HERE

Join us for TechMentor, August 6 – 8, 2018, as we return to Microsoft Headquarters in Redmond, WA. In today's IT world, more things change than stay the same. As we celebrate the 20th year of TechMentor, we are more committed than ever to providing immediately usable IT education, with the tools you need today, while preparing you for tomorrow – **keep up, stay ahead and avoid Winter, ahem, Change.**

Plus you'll be at the source, Microsoft HQ, where you can have lunch with Blue Badges, visit the company store, and experience life on campus for a week!

Are You Ready?

Hot Topics Covered:

- Windows PowerShell
- Windows Server
- DevOps
- Azure
- Hyper-V
- IT Security

EVENT PARTNER Microsoft

SUPPORTED BY Redmond · Redmond Channel Partner · Virtualization & Cloud Review

PRODUCED BY 1105 MEDIA — YOUR GROWTH, OUR BUSINESS.

In-Depth Training for IT Pros @ Microsoft Headquarters

You owe it to yourself, your company and your career to be at TechMentor Redmond 2018!

SAVE $400!
REGISTER NOW
Use Promo Code TMJAN2

techmentorevents.com/redmond

CONNECT WITH TECHMENTOR

Twitter
@TechMentorEvent

Facebook
Search "TechMentor"

LinkedIn
Search "TechMentor"

Creating a Line-of-Business App with the UWP

Bruno Sonnino

One day, your boss comes to you with a new assignment: You must create a new line-of-business (LOB) app for Windows. Web isn't an option and you must choose the best platform for it, to ensure it will be available for at least 10 more years.

That seems to be a difficult choice: You can select Windows Forms or Windows Presentation Foundation (WPF), which are mature technologies with a lot of components, and you, as a .NET developer, have a lot of knowledge and experience working with them. Will they be there in 10 years? My guess is yes; there's no sign that Microsoft will discontinue any of them in the near future. But in addition to being a bit old (Windows Forms is from 2001 and WPF was created in 2006), these technologies have some other problems: Windows Forms doesn't use the latest graphics cards and you're stuck with the same old boring style, unless you use extra components or do some magic tricks. WPF, though it's still being updated, hasn't caught up with the latest improvements to the OS and its SDK. Moreover, neither can be deployed to the Windows Store and benefit from it: There's no easy deploy and install/uninstall, worldwide discovery and distribution, and so on. You can use Centennial to package the apps, but that's not like getting "the real thing."

Digging a little more, you discover the Universal Windows Platform (UWP) for developing for Windows 10 and see that it doesn't have the downsides of Windows Forms and WPF. It's being actively improved and can be used with no change in a wide range of devices, from the tiny Raspberry Pi to the huge Surface Hub. But everybody says that the UWP is for developing small apps for Windows tablets and phones; there's no such thing as an LOB app using the UWP. And, besides that, it's very difficult to learn and develop: all those new patterns, the XAML language, the sandbox that limits what can be done, and on and on.

All that is far from the reality. Though in the past the UWP had some limitations, and could be difficult to learn and use, this is no longer true. There's been aggressive development of new features—with the new Fall Creators Update (FCU) and its SDK you can even use .NET Standard 2.0 APIs, as well as access SQL Server directly. At the same time, new tools and components can give you the best experience to develop a new app. Windows Template Studio (WTS) is an extension to Visual Studio that provides a fast start to

This article discusses:

- Universal Windows Platform development
- .NET Standard 2.0
- Windows Template Studio
- Telerik controls for the UWP

Technologies discussed:

Universal Windows Platform, .NET Standard 2.0, Windows Template Studio, SQL Server

Code download available at:

github.com/bsonnino/lobuwp

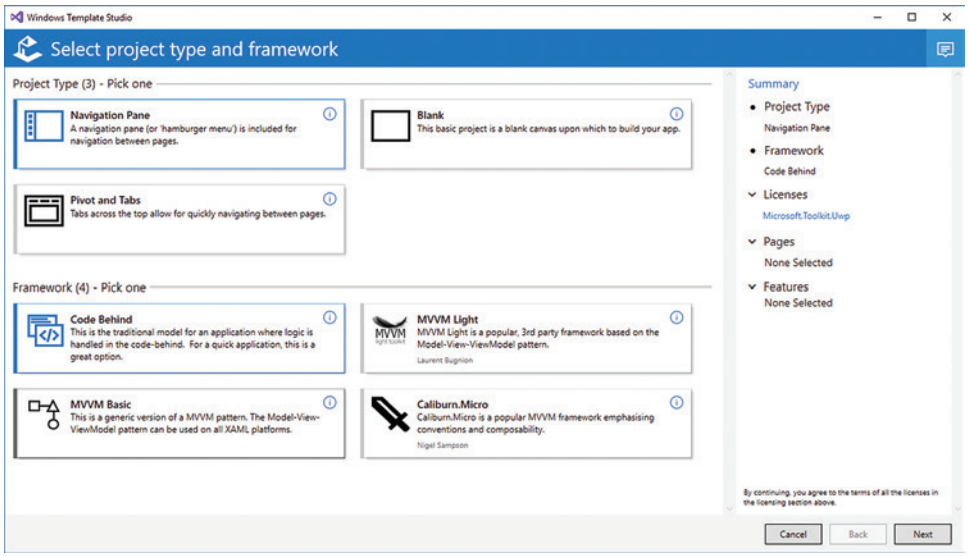


Figure 1 **Windows Template Studio Main Screen**

a full-featured UWP app and you can use the Telerik components for UWP for free, for the best UX. Not bad, no? Now you have a new platform for developing your UWP LOB apps!

In this article, I'm going to develop a UWP LOB app using WTS, the Telerik components and direct access to SQL Server using .NET Standard 2.0, so you can see how everything works firsthand.

Introduction to the Windows Template Studio

As I said before, you need the FCU and its corresponding SDK to use .NET Standard 2.0. Once you have FCU installed on your machine (if you aren't sure, press Win+R and type Winver to show your Windows version. Your version should be 1709 or newer). The support for .NET Standard 2.0 in the UWP is available only in Visual Studio 2017 update 4 or newer, so if you haven't updated, you must do it now. You also need the .NET 4.7 runtime installed to use it.

With the infrastructure in place, you can install WTS. In Visual Studio, go to Tools | Extensions and Updates and search for Windows Template Studio. After you download it, restart Visual Studio to finish the installation. You can also install it by going to aka.ms/wtsinstall, downloading the installer and then installing WTS.

Windows Template Studio is a new, open source extension (you can get the source code from aka.ms/wts) that will give you a great quick start to UWP development: You can choose the type of project (a Navigation Pane with the hamburger menu, Blank, or Pivot and Tabs), the Model-View-ViewModel (MVVM) framework for your MVVM pattern, the pages you want in your project and the Windows 10 features you want to include. There's a lot of development going on in this project, and PRISM, Visual Basic templates and Xamarin support are already available in the nightly builds (coming soon to the stable version).

With WTS installed, you can go to Visual Studio and create a new project using File | New Project | Windows Universal and select the Windows Template Studio. A new window will open to choose the project (see **Figure 1**).

Select the Navigation Pane project and the MVVM Light framework, then select Next. Now, choose the pages in the app, as shown in **Figure 2**.

As you can see in the right Summary column, there's already a blank page selected, named Main. Select a Settings page and name it Settings, and a Master-Detail page and name it Sales, then click on the Next button. In the next window you can select some features for the app, such as Live Tile or Toast, or a First Use dialog that will be presented the first time the app is run. I won't choose any features now, except the Settings storage that's selected when you choose the Settings page.

Now, click on the Create button to create the new app with the selected features. You can run the app—it's a complete app with a hamburger menu, two pages (blank and master-detail) and a settings page (accessed by clicking the cog icon at the bottom) that allows you to change themes for the app.

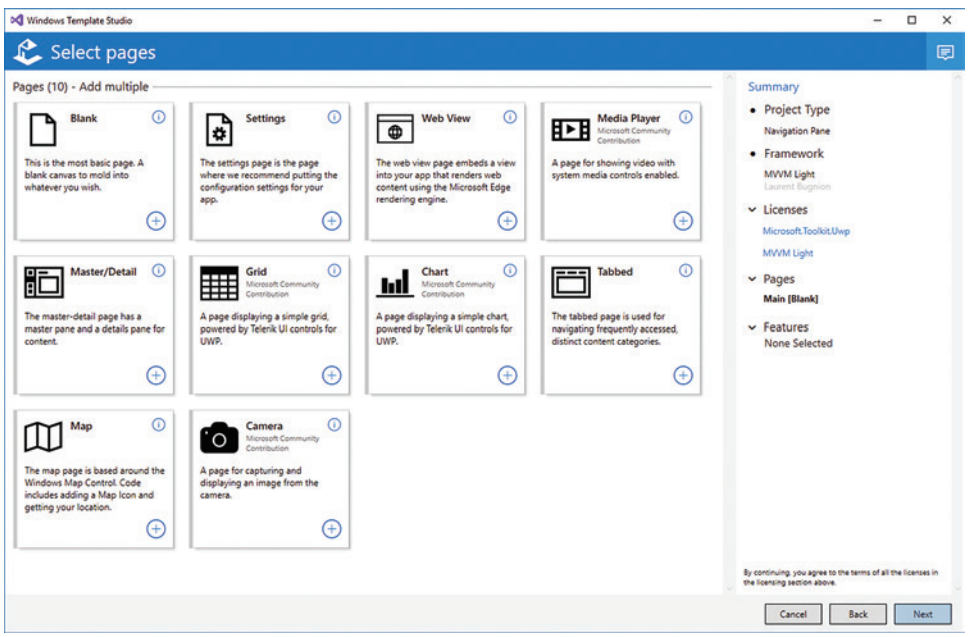


Figure 2 **Page Selection**

Figure 3 **The Order Class**

```
public class Order : INotifyPropertyChanged
{
  public long OrderId { get; set; }
  public DateTime OrderDate { get; set; }
  public string Company { get; set; }
  public decimal OrderTotal { get; set; }
  public DateTime? DatePicked { get; set; }
  public bool Delivered => DatePicked != null;
  private IEnumerable<OrderItem> _orderItems;
  public event PropertyChangedEventHandler PropertyChanged;
  public IEnumerable<OrderItem> OrderItems
  {
    get => _orderItems;

    set
    {
      _orderItems = value;
      PropertyChanged?.Invoke(
        this, new PropertyChangedEventArgs("OrderItems"));
    }
  }

  public override string ToString() => $"{Company} {OrderDate:g}  {OrderTotal}";
}
```

If you go to Solution Explorer, you'll see that many folders were created for the app, such as Models, ViewModels and Services. There's even a Strings folder with an en-US subfolder for the localization of strings. If you want to add more languages to the app, you just need to add a new subfolder to the Strings folder, name it with the locale of the language (like fr-FR), copy the Resources.resw file and translate it to the new language. Impressive what you've been able to do with just a few clicks, no? But I'm pretty sure that's not what your boss had in mind when he gave you the assignment, so let's go customize that app!

Accessing SQL Server from the App

One nice feature that was introduced in the FCU and in Visual Studio update 4 is the support for .NET Standard 2.0 in the UWP. This is a great improvement, because it allows UWP apps to use a

Figure 4 **SalesViewModel**

```
public class SalesViewModel : ViewModelBase
{
  private Order _selected;

  public Order Selected
  {
    get => _selected;
    set
    {
      Set(ref _selected, value);
    }
  }

  public ObservableCollection<Order> Orders { get; private set; }

  public async Task LoadDataAsync(MasterDetailsViewState viewState)
  {
    var orders = await DataService.GetOrdersAsync();
    if (orders != null)
    {
      Orders = new ObservableCollection<Order>(orders);
      RaisePropertyChanged("Orders");
    }
    if (viewState == MasterDetailsViewState.Both)
    {
      Selected = Orders.FirstOrDefault();
    }
  }
}
```

Figure 5 **Code to Retrieve the Orders from the Database**

```
public static async Task<IEnumerable<Order>> GetOrdersAsync()
{
  using (SqlConnection conn = new SqlConnection(
    "Database=WideWorldImporters;Server=.;User ID=sa;Password=pass"))
  {
    try
    {
      await conn.OpenAsync();
      SqlCommand cmd = new SqlCommand("select o.OrderId, " +
        "c.CustomerName, o.OrderDate, o.PickingCompletedWhen, " +
        "sum(l.Quantity * l.UnitPrice) as OrderTotal " +
        "from Sales.Orders o " +
        "inner join Sales.Customers c on c.CustomerID = o.CustomerID " +
        "inner join Sales.OrderLines l on o.OrderID = l.OrderID " +
        "group by o.OrderId, c.CustomerName, o.OrderDate,
          o.PickingCompletedWhen " +
        "order by o.OrderDate desc", conn);

      var results = new List<Order>();
      using (SqlDataReader reader = await cmd.ExecuteReaderAsync())
      {
        while(reader.Read())
        {
          var order = new Order
          {
            Company = reader.GetString(1),
            OrderId = reader.GetInt32(0),
            OrderDate = reader.GetDateTime(2),
            OrderTotal = reader.GetDecimal(4),
            DatePicked = !reader.IsDBNull(3) ? reader.GetDateTime(3) :
              (DateTime?)null
          };
          results.Add(order);
        }
        return results;
      }
    }
    catch
    {
      return null;
    }
  }
}

public static async Task<IEnumerable<OrderItem>> GetOrderItemsAsync(
  int orderId)
{
  using (SqlConnection conn = new SqlConnection(
        "Database=WideWorldImporters;Server=.;User ID=sa;Password=pass"))
  {
    try
    {
      await conn.OpenAsync();
      SqlCommand cmd = new SqlCommand(
        "select Description,Quantity,UnitPrice " +
        $"from Sales.OrderLines where OrderID = {orderId}", conn);

      var results = new List<OrderItem>();
      using (SqlDataReader reader = await cmd.ExecuteReaderAsync())
      {
        while (reader.Read())
        {
          var orderItem = new OrderItem
          {
            Description = reader.GetString(0),
            Quantity = reader.GetInt32(1),
            UnitPrice = reader.GetDecimal(2),
          };
          results.Add(orderItem);
        }
        return results;
      }
    }
    catch
    {
      return null;
    }
  }
}
```

Spreadsheets Made Easy.

SpreadsheetGear 2017 Released

SpreadsheetGear 2017 adds a new SpreadsheetGear for .NET Standard product, official support for Excel 2013 and Excel 2016, 51 new Excel functions for a total of 449 fully supported functions, full conditional formatting support, enhanced workbook protection and encryption, cell gradient rendering and more.

Scalable Reporting

Easily create richly formatted Excel reports without Excel from any ASP.NET, Windows Forms, WPF or Silverlight application using spreadsheet technology built from the ground up for performance, scalability and reliability.

.NET Standard

Support for iOS, Android, Linux, macOS, UWP and more

SpreadsheetGear for .NET Standard enables cross-platform developers to enjoy the same high performance Excel-compatible reporting, charting, calculations and more relied on by thousands of Windows developers for 10+ years.

WIN Windows Forms | Silverlight | WPF WPF

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.

Fastest Calculations

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

Download your free fully functional evaluation at SpreadsheetGear.com

SpreadsheetGear

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

huge number of APIs that were unavailable before, including SQL Server client access and Entity Framework Core.

To make SQL Server client access available in your app, go to the app properties and select Build 16299 as the Minimum Version in the application tab. Then right-click the References node and select "Manage NuGet packages" and install System.Data.SqlClient.

Figure 6 **The List Items Data Template**

```
<DataTemplate x:Key="ItemTemplate" x:DataType="model:Order">
  <Grid Height="64" Padding="0,8">
    <Grid.ColumnDefinitions>
      <ColumnDefinition Width="Auto"/>
      <ColumnDefinition Width="*"/>
    </Grid.ColumnDefinitions>
    <StackPanel Grid.Column="1" Margin="12,0,0,0" VerticalAlignment="Center">
      <TextBlock Text="{x:Bind Company}" Style="{ThemeResource ListTitleStyle}"/>
      <StackPanel Orientation="Horizontal">
        <TextBlock Text="{x:Bind OrderDate.ToShortDateString()}"
                   Margin="0,0,12,0" />
        <TextBlock Text="{x:Bind OrderTotal}" Margin="0,0,12,0" />
      </StackPanel>
    </StackPanel>
  </Grid>
</DataTemplate>
<DataTemplate x:Key="DetailsTemplate">
  <views:SalesDetailControl MasterMenuItem="{Binding}"/>
</DataTemplate>
```

Figure 7 **Changes in SalesDetailControl**

```
<Grid Name="block" Padding="0,15,0,0">
  <Grid.Resources>
    <Style x:Key="RightAlignField" TargetType="TextBlock">
      <Setter Property="HorizontalAlignment" Value="Right" />
      <Setter Property="VerticalAlignment" Value="Center" />
      <Setter Property="Margin" Value="0,0,12,0" />
    </Style>
  </Grid.Resources>
  <Grid.RowDefinitions>
    <RowDefinition Height="Auto" />
    <RowDefinition Height="Auto" />
    <RowDefinition Height="*" />
  </Grid.RowDefinitions>
  <TextBlock Margin="12,0,0,0"
    Text="{x:Bind MasterMenuItem.Company, Mode=OneWay}"
    Style="{StaticResource SubheaderTextBlockStyle}" />
  <StackPanel Orientation="Horizontal" Grid.Row="1" Margin="12,12,0,12">
    <TextBlock Text="Order date:"
      Style="{StaticResource BodyTextBlockStyle}" Margin="0,0,12,0"/>
    <TextBlock Text="{x:Bind MasterMenuItem.OrderDate.ToShortDateString(),
      Mode=OneWay}"
      Style="{StaticResource BodyTextBlockStyle}" Margin="0,0,12,0"/>

    <TextBlock Text="Order total:" Style="{StaticResource BodyTextBlockStyle}"
      Margin="0,0,12,0"/>
    <TextBlock Text="{x:Bind MasterMenuItem.OrderTotal, Mode=OneWay}"
      Style="{StaticResource BodyTextBlockStyle}" />
  </StackPanel>
  <grid:RadDataGrid ItemsSource="{x:Bind MasterMenuItem.OrderItems,
    Mode=OneWay}" Grid.Row="2"
    UserGroupMode="Disabled" Margin="12,0"
    UserFilterMode="Disabled" BorderBrush="Transparent"
      AutoGenerateColumns="False">
    <grid:RadDataGrid.Columns>
      <grid:DataGridTextColumn PropertyName="Description" />
      <grid:DataGridNumericalColumn
        PropertyName="UnitPrice"  Header="Unit Price"
        CellContentStyle="{StaticResource RightAlignField}"/>
      <grid:DataGridNumericalColumn
        PropertyName="Quantity"  Header="Quantity"
        CellContentStyle="{StaticResource RightAlignField}"/>
      <grid:DataGridNumericalColumn
        PropertyName="TotalPrice"  Header="Total Price"
        CellContentStyle="{StaticResource RightAlignField}"/>
    </grid:RadDataGrid.Columns>
  </grid:RadDataGrid>
</Grid>
```

With that, you're ready to access a local SQL Server database. One note: the access isn't made by using named pipes—the default access method—but by using TCP/IP. So, you must run the SQL Server Configuration app and enable TCP/IP connections for your server instance.

I'll also use the Telerik UI for UWP grid, now a free open source product that you'll find at bit.ly/2AFWktT. So, while you still have the NuGet package manager window open, select and install the Telerik.UI.for.UniversalWindowsPlatform package. If you're adding a grid page in WTS, this package is installed automatically.

For this app, I'll use the WorldWideImporters sample database, which you can download from bit.ly/2fLYuBk and restore it to your SQL Server instance.

Now, you must change the default data access. If you go to the Models folder, you'll see the SampleOrder class, with this comment at the beginning:

```
// TODO WTS: Remove this class once your pages/features are using your data.
// This is used by the SampleDataService.
// It is the model class we use to display data on pages like Grid, Chart,
and Master Detail.
```

There are a lot of comments throughout the project that give guidance on what you need to do. In this case, you need an Order class that's very similar to this one. Rename the class to Order and change it to something similar to the one in **Figure 3**.

This class implements the INotifyPropertyChanged interface, because I want to notify the UI when the order items are changed so I can load them on demand when displaying the order. I've defined another class, OrderItem, to store the order items:

```
public class OrderItem
{
  public string Description { get; set; }
  public decimal UnitPrice { get; set; }
  public int Quantity { get; set; }
  public decimal TotalPrice => UnitPrice * Quantity;
}
```

The SalesViewModel, shown in **Figure 4**, must also be modified to reflect these changes.

When the Selected property is changed, it will check whether the order items are already loaded and, if not, it calls the GetOrderItemsAsync method in the data service to load them.

The last change in the code to make is in the SampleDataService class, to remove the sample data and create the SQL Server access, as shown in **Figure 5**. I've renamed the class to DataService, to reflect that it's not a sample anymore.

The code is the same you'd use in any .NET app; there's no change at all. Now I need to modify the list items data template in SalesPage.xaml to reflect the changes, as shown in **Figure 6**.

I need to change the DataType so it refers to the new Order class and change the fields presented in the TextBlocks. I must also change the codebehind in SalesDetailControl.xaml.cs to load the Items for the selected order when it's changed. This is done in the OnMasterMenuItemChanged method, which is transformed to an async method:

```
private static async void OnMasterMenuItemChangedAsync(DependencyObject d,
  DependencyPropertyChangedEventArgs e)
{
  var newOrder = e.NewValue as Order;
  if (newOrder != null && newOrder.OrderItems == null)
    newOrder.OrderItems = await
      DataService.GetOrderItemsAsync((int)newOrder.OrderId);
}
```

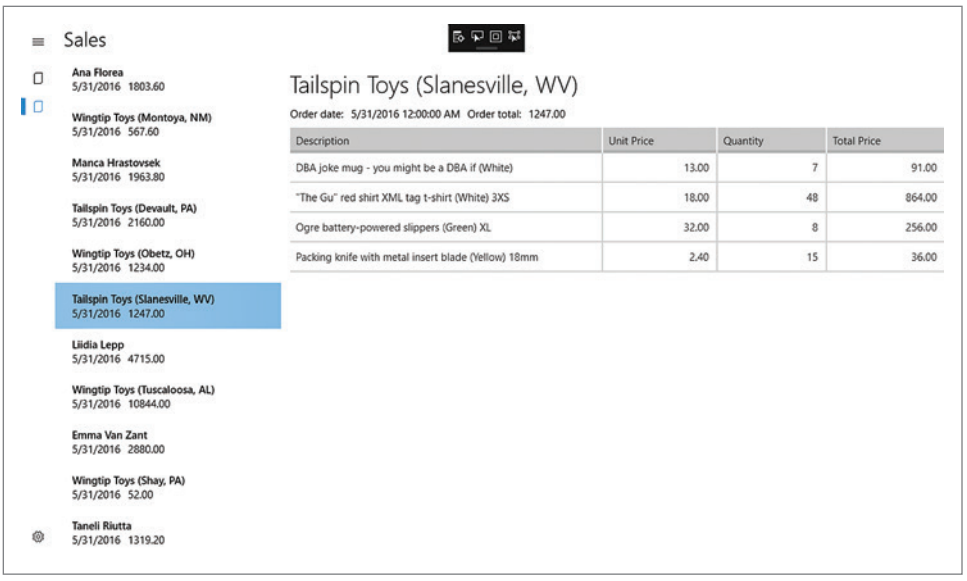


Figure 8 **Application Showing Database Orders**

Next, I must change SalesDetailControl to point to the new fields and show them, as shown in **Figure 7**.

I display the sales data in the top of the control and use a Telerik RadDataGrid to show the order items. Now, when I run the application, I get the orders on the second page, like what's shown in **Figure 8**.

I still have the empty main page. I'll use it to show a grid of all customers. In MainPage.xaml, add the DataGrid to the content grid:

```
<grid:RadDataGrid ItemsSource="{Binding Customers}"/>
```

You must add the namespace to the Page tag, but you don't need to remember the syntax and correct namespace. Just put the mouse cursor over RadDataGrid, type Ctrl+. and a box will open, indicating the right namespace to add. As you can see from the added line, I'm binding the ItemsSource property to Customers. The DataContext for the page is an instance of MainViewModel, so I must create this property in MainViewModel.cs, as shown in **Figure 9**.

I'm loading the customers when the ViewModel is created. One thing to notice here is that I don't wait for the loading to complete. When the data is fully loaded, the ViewModel indicates that the Customers property has changed and that loads the data in the view. The GetCustomersAsync method in DataService is very similar to GetOrdersAsync, as you can see in **Figure 10**.

Figure 9 **MainViewModel Class**

```
public class MainViewModel : ViewModelBase
{
  public ObservableCollection<Customer> Customers { get; private set; }

  public async Task LoadCustomersAsync()
  {
    var customers = await DataService.GetCustomersAsync();
    if (customers != null)
    {
      Customers = new ObservableCollection<Customer>(customers);
      RaisePropertyChanged("Customers");
    }
  }
  public MainViewModel()
  {
    LoadCustomersAsync();
  }
}
```

With that, you can run the app and see the customers on the main page, as shown in **Figure 11**. Using the Telerik grid, you get a lot of goodies for free: grouping, sorting and filtering are built into the grid and there's no extra work to do.

Finishing Touches

I have now a basic LOB application, with a customers grid and a master-detail view showing the orders, but it can use some finishing touches. The icons in the lateral bar for both pages are the same and they can be customized. These icons are set in the ShellViewModel. If you go there, you'll see these comments, which point to where to go to change the icons and the text for the items:

```
// TODO WTS: Change the symbols for each item as appropriate for your app
// More on Segoe UI Symbol icons:
// https://docs.microsoft.com/windows/uwp/style/segoe-ui-symbol-font
// Or to use an IconElement instead of a Symbol see
// https://github.com/Microsoft/WindowsTemplateStudio/blob/master/docs/
projectTypes/navigationpane.md
// Edit String/en-US/Resources.resw: Add a menu item title for each page
```

You can use font symbol icons (as in the actual code) or images from other sources if you use IconElements. (You can use .png files, XAML paths, or characters from any other fonts; take a look at bit.ly/2zICuB2 for more information.) I'm going to use two symbols from Segoe UI Symbol, People and ShoppingCart. To do that, I must change the NavigationItems in the code:

```
_primaryItems.Add(new ShellNavigationItem("Shell_Main".GetLocalized(),
  Symbol.People, typeof(MainViewModel).FullName));
_primaryItems.Add(new ShellNavigationItem("Shell_Sales".GetLocalized(),
  (Symbol)0xE7BF, typeof(SalesViewModel).FullName));
```

For the first item, there's already a symbol in the Symbol enumeration, Symbol.People, but for the second there's no such enumeration, so I use the hex value and cast it to the Symbol enumeration. To change the title of the page and the caption of the menu item, I edit Resources.resw and change Shell_Main and Main_Title.Text to Customers. I can also add some customization to the grid by changing some properties:

```
<grid:RadDataGrid ItemsSource="{Binding Customers}"
UserColumnReorderMode="Interactive"
                ColumnResizeHandleDisplayMode="Always"
                AlternationStep="2" AlternateRowBackground="LightBlue"/>
```

Adding a Live Tile to the App

I can also enhance the application by adding a live tile. I'll do so by going to Solution Explorer, right-clicking the project node and selecting Windows Template Studio | New feature, then choosing Live Tile. When you click on the Next button, it will show the affected tiles (both new ones and those that have been changed), so you can see if you really like what you did. Clicking on the Finish button will add the changes to the app.

Everything to add a Live Tile will be there, you just need to create the tile content. This is already done in LiveTileService.Samples. It's a partial class that adds the method SampleUpdate to

Figure 10 **GetCustomersAsync Method**

```
public static async Task<IEnumerable<Customer>> GetCustomersAsync()
{
  using (SqlConnection conn = new SqlConnection(
    "Database=WideWorldImporters;Server=.;User ID=sa;Password=pass"))
  {
    try
    {
      await conn.OpenAsync();
      SqlCommand cmd = new SqlCommand("select c.CustomerID,
        c.CustomerName, " +
        "cat.CustomerCategoryName, c.DeliveryAddressLine2,
          c.DeliveryPostalCode, " +
        "city.CityName, c.PhoneNumber " +
        "from Sales.Customers c " +
        "inner join Sales.CustomerCategories cat on c.CustomerCategoryID =
          cat.CustomerCategoryID " +
        "inner join Application.Cities city on c.DeliveryCityID =
          city.CityID", conn);

      var results = new List<Customer>();
      using (SqlDataReader reader = await cmd.ExecuteReaderAsync())
      {
        while (reader.Read())
        {
          var customer = new Customer
          {
            CustomerId = reader.GetInt32(0),
            Name = reader.GetString(1),
            Category = reader.GetString(2),
            Address = reader.GetString(3),
            PostalCode = reader.GetString(4),
            City = reader.GetString(5),
            Phone = reader.GetString(6)
          };
          results.Add(customer);
        }
        return results;
      }
    }
    catch
    {
      return null;
    }
  }
}
```

LiveTileService. As shown in **Figure 12**, I'll rename the file to LiveTileService.LobData and add two methods to it, UpdateCustomerCount and UpdateOrdersCount, which will show in the live tiles how many customers or orders are in the database.

UpdateSample was originally called in the initialization, in the StartupAsync method of ActivationService. I'm going to replace it with the new UpdateCustomerCount:

```
private async Task StartupAsync()
{
  Singleton<LiveTileService>.Instance.UpdateCustomerCount(0);
  ThemeSelectorService.SetRequestedTheme();
  await Task.CompletedTask;
}
```

At this point, I still don't have the customer count to update. That will happen when I get the customers in MainViewModel:

```
public async Task LoadCustomersAsync()
{
  var customers = await DataService.GetCustomersAsync();
  if (customers != null)
  {
    Customers = new ObservableCollection<Customer>(customers);
    RaisePropertyChanged("Customers");
    Singleton<LiveTileService>.Instance.UpdateCustomerCount(Customers.Count);
  }
}
```

The orders count will be updated when I get the orders in SalesViewModel:

```
public async Task LoadDataAsync(MasterDetailsViewState viewState)
{
  var orders = await DataService.GetOrdersAsync();
  if (orders != null)
  {
    Orders = new ObservableCollection<Order>(orders);
    RaisePropertyChanged("Orders");
    Singleton<LiveTileService>.Instance.UpdateOrderCount(Orders.Count);
  }
  if (viewState == MasterDetailsViewState.Both)
  {
    Selected = Orders.FirstOrDefault();
  }
}
```

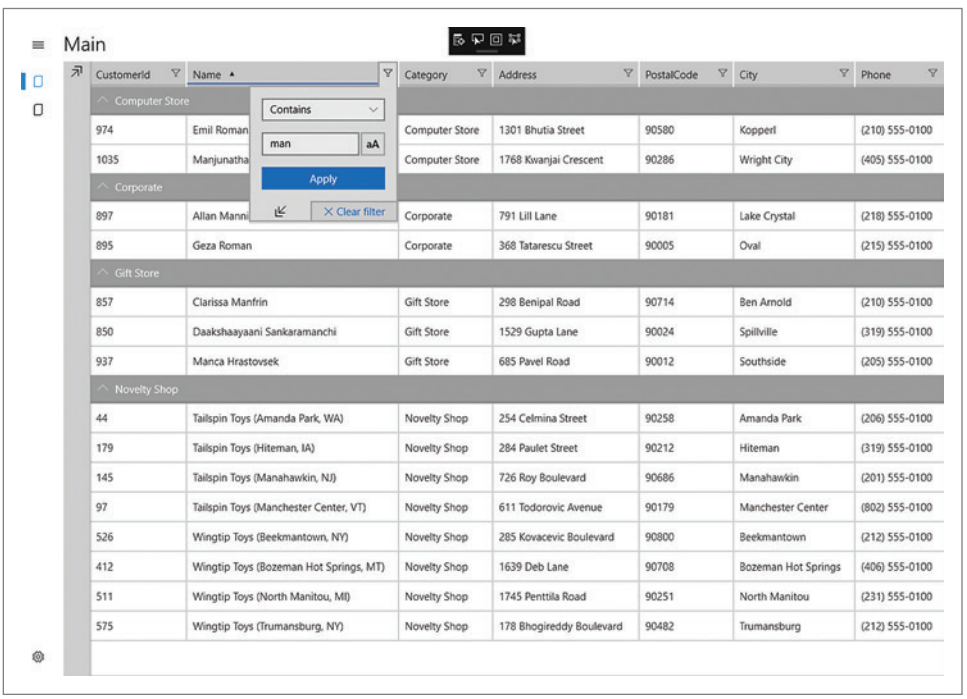


Figure 11 **Customers Grid with Grouping and Filtering**

With that, I have an app as shown in **Figure 13**.

This app can show the customers and orders retrieved from a local database, updating the live tile with the customers and order counts. You can group, sort or filter the customers listed and show the orders using a master-detail view. Not bad!

Wrapping Up

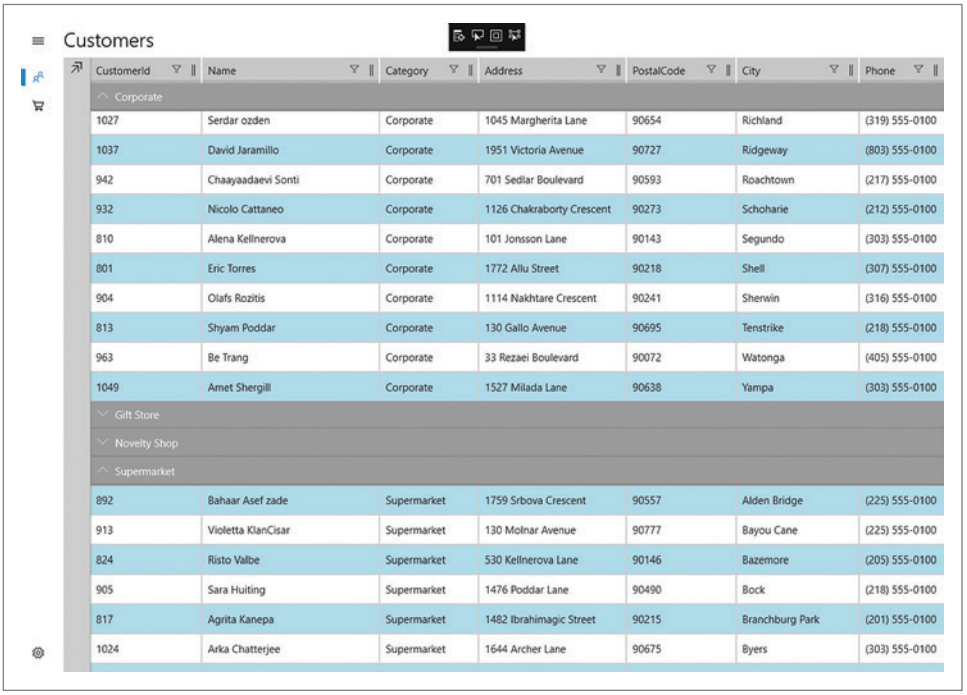
As you can see, the UWP isn't just for small apps. You can use it for your LOB apps, getting the data from many sources, including a local SQL Server (and you can even use Entity Framework as an ORM). With .NET Standard 2.0, you have access to a lot of APIs that are already available in the .NET Framework, with no change. WTS gives you a quick start, helping you to quickly and easily create an application using

Figure 12 **Class to Update the Live Tile**

```
internal partial class LiveTileService
{
  private const string TileTitle = "LoB App with UWP";

  public void UpdateCustomerCount(int custCount)
  {
    string tileContent =
      $@"There are {(custCount > 0 ? custCount.ToString() : "no")}
        customers in the database";
    UpdateTileData(tileContent, "Customer");
  }

  public void UpdateOrderCount(int orderCount)
  {
    string tileContent =
      $@"There are {(orderCount > 0 ? orderCount.ToString() : "no")}
        orders in the database";
    UpdateTileData(tileContent,"Order");
  }

  private void UpdateTileData(string tileBody, string tileTag)
  {
    TileContent tileContent = new TileContent()
    {
      Visual = new TileVisual()
      {
        TileMedium = new TileBinding()
        {
          Content = new TileBindingContentAdaptive()
          {
            Children =
            {
              new AdaptiveText()
              {
                Text = TileTitle,
                HintWrap = true
              },
              new AdaptiveText()
              {
                Text = tileBody,
                HintStyle = AdaptiveTextStyle.CaptionSubtle,
                HintWrap = true
              }
            }
          }
        },

        TileWide = new TileBinding()
        {
          Content = new TileBindingContentAdaptive()
          {
            Children =
            {
              new AdaptiveText()
              {
                Text = $"{TileTitle}",
                HintStyle = AdaptiveTextStyle.Caption
              },
              new AdaptiveText()
              {
                Text = tileBody,
                HintStyle = AdaptiveTextStyle.CaptionSubtle,
                HintWrap = true
              }
            }
          }
        }
      }
    };
    var notification = new TileNotification(tileContent.GetXml())
    {
      Tag = tileTag
    };

    UpdateTile(notification);
  }
}
```

best practices with the tools you prefer, and to add Windows features to it. There are great UI controls to enhance the appearance of the app and the app will run in a wide range of devices: on a phone, a desktop, the Surface Hub and even on HoloLens without change.

When it comes to deploying, you can send your app to the store and have worldwide discovery, easy install and uninstall and auto updates. If you don't want to deploy it through the store, you can do so using the Web (bit.ly/2zHOnZY). As you can see, when you have to create a new LOB app for Windows, you should certainly consider the UWP, because it can offer you everything you want for this kind of app and more, with the great advantage that it's being aggressively developed and will bring you a lot of improvements for years to come. ■



CustomerId	Name	Category	Address	PostalCode	City	Phone
Corporate						
1027	Serdar ozden	Corporate	1045 Margherita Lane	90654	Richland	(319) 555-0100
1037	David Jaramillo	Corporate	1951 Victoria Avenue	90727	Ridgeway	(803) 555-0100
942	Chaayaadaevi Sonti	Corporate	701 Sedlar Boulevard	90593	Roachtown	(217) 555-0100
932	Nicolo Cattaneo	Corporate	1126 Chakraborty Crescent	90273	Schoharie	(212) 555-0100
810	Alena Kellnerova	Corporate	101 Jonsson Lane	90143	Segundo	(303) 555-0100
801	Eric Torres	Corporate	1772 Allu Street	90218	Shell	(307) 555-0100
904	Olafs Rozitis	Corporate	1114 Nakhtare Crescent	90241	Sherwin	(316) 555-0100
813	Shyam Poddar	Corporate	130 Gallo Avenue	90695	Tenstrike	(218) 555-0100
963	Be Trang	Corporate	33 Rezaei Boulevard	90072	Watonga	(405) 555-0100
1049	Amet Shergill	Corporate	1527 Milada Lane	90638	Yampa	(303) 555-0100
Gift Store						
Novelty Shop						
Supermarket						
892	Bahaar Asef zade	Supermarket	1759 Srbova Crescent	90557	Alden Bridge	(225) 555-0100
913	Violetta KlanCisar	Supermarket	130 Molnar Avenue	90777	Bayou Cane	(225) 555-0100
824	Risto Valbe	Supermarket	530 Kellnerova Lane	90146	Bazemore	(205) 555-0100
905	Sara Huiting	Supermarket	1476 Poddar Lane	90490	Bock	(218) 555-0100
817	Agrita Kanepa	Supermarket	1482 Ibrahimagic Street	90215	Branchburg Park	(201) 555-0100
1024	Arka Chatterjee	Supermarket	1644 Archer Lane	90675	Byers	(303) 555-0100

Figure 13 **The Finished App**

Bruno Sonnino *has been a Microsoft Most Valuable Professional (MVP) since 2007. He's a developer, consultant, and author, having written many books and articles about Windows development. You can follow him on Twitter: @bsonnino or read his blog posts at blogs.msmvps.com/bsonnino.*

Thanks to the following Microsoft technical expert who reviewed this article: *Clint Rutkas*

20 Years of Cutting Edge: A Conversation

A lot has changed in the IT world since the mid-1990s, yet *MSDN Magazine*—or more accurately, its progenitors *Microsoft Systems Journal* (MSJ) and *Microsoft Interactive Developer* (MIND)—were there, informing developers of the latest tools, techniques and technologies they needed to get ahead in the world of Windows programming.

The idea of writing for either of these publications was at the time a dream. I had always been fascinated by writing, going back to my high school years in a small beach-front home on the eastern coast of central Italy. In an era where e-mail seemed to suddenly give everybody the chance to talk to nearly anyone, I worked up the courage to propose technical articles to prestigious magazines like *Dr. Dobb's Journal*. By 1996 I was a published author, but I still hadn't written for either Microsoft magazine.

That all changed when I came across a column called "The Visual Programmer" in MSJ. At the time I was the epitome of the "real" programmer, doing pointer-based stuff in raw C without even the thin protections of the emerging C++ language. I used to look with suspicion on things like Microsoft Foundation Classes (MFC), which seemed designed to shield lazy developers from the harsh realities of programming. I didn't think highly of those doing "visual" development.

Then, in the October 1996 issue of MSJ, I read a column that boldly claimed to teach readers the way to add VBScript to their existing apps. The author, Josh Trupin, started by apologizing for having missed the last couple issues due to a new role he'd taken at Microsoft. "If you've written to me and I haven't sent you a reply, it's not because I'm ignoring you. It's because I'm busy and I'm ignoring you," he wrote.

I was impressed with the article, but felt also that I was every bit as sharp as the author. If he could write a column for MSJ, why not me? So I contacted Josh to praise his work, only to learn that he had just assumed the role of technical editor at MIND. It didn't take me long to offer to write for him.


Interactive Developer

Contents

VOLUME 2 NUMBER SIX • JUNE 1997 • www.microsoft.com/mind/

14 Design-Time ActiveX Controls Made Easy with Visual C++ 5.0
By Steve Zimmerman
Design-time ActiveX controls are used during Web site development to generate embedded text without sending a control across the Web. With Visual C++ 5.0, these controls can be created easily and quickly.

24 Supporting CryptoAPI in Real-World Applications
By Dino Esposito
CryptoAPI is an interface developed by Microsoft to provide any application, not just Web-based programs, with easy-to-implement security features.

34 A Brief History of Digital Security and Cryptography
By Brian Jamison, Josh Gold, and Warren Jamison
Although transaction security on the Internet is a hot topic, it's only part of the colorful history of cryptography.

40 Active Messaging and the Microsoft Exchange Server
By Dan Fay and Thomas Rizzo
With Microsoft Exchange Server, you can create workgroup-enabled applications that run across the Web—right in your Web browser.

48 Integrating the Microsoft Index Server with Active Server Pages
By Christian Gross
A standard part of Internet Information Server 3.0, the Microsoft Index Server lets you catalog documents on your system. A simple query page can quickly unlock the power of the server for your Web site.

72 The Site Building Series Part III: The Software on the Server
By Jason Coombs, Ted Coombs, and John Hovis
In Part III of the Site Building Series, we get down to specifics—the software you use to set up an Internet site.

PRINTED IN USA

AWESOME INTERNET SERVICES

4 Editor's Note
6 Contributors
8 Mail
10 New Stuff
80 Geektogeek
84 How It's Done
88 Java 911
96 Advertisers Index
98 Flux

Figure 1 **Dino's First Article in the Contents Page of *Microsoft Interactive Developer***

The Inception of a Digital Friendship

Josh and I started collaborating right after that, and we've remained friends ever since. The first article I wrote for him was published in the June 1997 issue of MIND (**Figure 1**). A few months later in January I published what would be the first of many Cutting Edge columns, initially in MIND and later in *MSDN Magazine*.

Now, exactly 20 years after my first Cutting Edge installment, I wanted to take a moment to look back at the last two decades. And what better way to do that, than to rope in my old friend and colleague Josh Trupin to provide some perspective. Here's our conversation.

Dino Esposito: Josh, do you remember my first article in the June 1997 issue of MIND?

Josh Trupin: I remember you sending me so many e-mails that I couldn't just ignore you. But, yes, I really enjoyed your first article ever. What was it? CryptoAPI, right?

Dino: Right. I still have a paper copy of the magazine. I liked it so much (well, I especially liked the check) that I wanted to get working on a second article right away. But getting a response from you was hard! You then told me you had been lame on replies because of your dentist.

Josh: Really? Well, you were more painful than him.

Dino: That's why you gave me the Cutting Edge column?

Josh: Not really. John Grieb had just started writing a new column called Cutting Edge in the November 1997 issue, but after the first article he resigned and I needed to find a replacement. Giving you the column solved two problems with a single move.

Dino: My first column dates back to January 1998. I think it was on something called Active Scripting.

Josh: Your memory is better than mine. You're really able to recall that?

Dino: Well, 1998 was a memorable year. In February I joined the first big company in my career. In May my son Francesco was born. In September I left the last big company in my career and decided that I could spend my life writing and coding.

Josh: Your son should be a man now. Does he do any programming?

Dino: He does, actually. He wrote his first mobile app when he was 12 years old, for Windows Phone. He even got a free device from Microsoft.

Josh: Amazing!

Dino: Well, I've been doing pair programming for quite some time, and with some success I'd add. You see the picture of my son playing on the computer (**Figure 2**)? There's an old copy of MIND in there about e-mails. I was actually reading that magazine cover to cover. I always had it around.

Josh: Does Francesco have more programs planned for the years to come?

Dino: He has a math mind and loves esoteric things like neural networks and quantum computing. To my great surprise, the other day he mentioned that he has two goals at this stage of his life. One is to meet Scott Guthrie in person. He can't believe that a demigod like a Microsoft vice president could really have any memory of me. He also can't believe that there was a time when I was more popular than Scott! I met Scott for the first time back in 1999 when he was unveiling an antediluvian thing called ASP+.

Josh: And the other desire?

Dino: Joining Michael Freedman's team at Microsoft.

Josh: Awesome. And what's that, exactly?

Dino: Michael Freedman is the brains behind the quantum computing effort at Microsoft.

Josh: I heard about Microsoft's quantum computing work at the Ignite conference, but it seems rather futuristic to me.

Dino: Well, I find it excitedly scary. For example, quantum computing could make affordable exponential calculations that today guarantee data privacy in crucial transactions. Have you ever realized that most of our stuff is secured by the unknown binding between a pair of lovely prime numbers?

Josh: Call me romantic, but I wish I could go back to the early days of scripting.

Dino: You could do a lot of JavaScript today!

Josh: Yeah, but even JavaScript is no longer the thing it was 20 years ago. But it did manage to survive while scary three-letter acronyms like OLE and COM went the way of the dinosaurs. What was the weirdest thing you wrote about in your 20 years of columns?

Dino: Well, in 20 years there have been plenty of technologies I covered that in the end were fated not to last. One I recall is ActiveX Documents, the technology to edit Word and other files in a browser. Another is Silverlight, which was furiously evangelized for a few years before being dropped. Perhaps worst of all, I repeatedly assured people that ASP.NET themes were about to overtake CSS stylesheets in a very short time. Oops.

Josh: You survived. I mean, 20 years of technical writing is a long time. We were two old men of 30 back in 1998, and we're two old men of 50 today. What do you see today in front of the current generation of old men of 30?

Dino: Interesting question. I see the next decade as the comeback of algorithms and modeling over pure technology and tools. Look at artificial intelligence (AI), for example. Sure, Microsoft is giving us wonderful tools like bots and Cognitive Services, and maybe enterprise-level Blockchain protocols in the near future. To make AI in the real world, to make it scale from the level of cool demos and articles to the real world, we need to learn about problems and problem domains. We need to learn how to build effective AI architectures, which, in essence, is understanding and modeling problems onto abstract structures. It seems to be an executive summary of the operational research exam when I was at college.

Today, we have big amounts of data but only run raw, stupid, brute-force algorithms on it. We have neural networks that still largely rely on the principles of Bayesian statistics, which were formulated back in the 18th century, over two centuries ago. We have a lot ahead of us, but most of it is still hidden.

I see the next decade as the comeback of algorithms and modeling over pure technology and tools.

Josh: Let's get back to planet Earth. How is your family? You also have a daughter, right?

Dino: The funny thing is that I got married when Microsoft released Windows 95, had my first son when Windows 98 was on its way, and our daughter was born around the time that Windows 2000 shipped.

Figure 2 **My Dad Is Not That Good with Computers**

Josh: So you're on Linux now or you're still upgrading your editions of Windows?

Dino: Haha. I just disabled automatic updates. No more kids for me, thanks.

Josh: Me too.

Dino: When was the last time we met? It's been a "long time, no see" kind of thing.

Josh: We didn't meet more than two or three times, I think. Then I left the magazine and a lot has happened since. But it's great to see that *MSDN Magazine* is still alive and kicking, putting out great content, even in the era of blogs and StackOverflow and Google.

Dino: When we started the Cutting Edge column, digital photography was in its infancy, Google was in beta, and smartphones were the stuff of science fiction. At the same time, the amount of knowledge needed to be a good professional has grown unbelievably. One could spend the weekend in the office—I did it a few times—bent over a collection of MSDN CDs, and on Monday be up-to-date on the state-of-the-art of Windows technology.

Today's knowledge base is like CosmosDB compared to Microsoft Access. The way developers gain access to technical information has changed, but being quick-to-find information is not the same as learning, or getting acquainted with, a new technology or a framework.

Josh: Is this the reason why you still write articles?

Dino: I write because I love writing. It also helps that I probably have a knack for abstracting core facts and concepts and conveying them in compelling and useful ways.

Josh: What was your best article ever?

Dino: I can't recall all the articles I wrote, but I remember the days of ASP.NET 2.0 as being very exciting. That was during the first half of the 2000s. More recently, I loved writing about event sourcing and CQRS (msdn.com/magazine/mt185569).

Josh: What will you never forget from these 20 years of Cutting Edge?

Dino: The e-mails we exchanged the morning (actually, my afternoon) of 9/11. I was not watching TV, just listening to the radio and one of your e-mails told me a couple of minutes before the radio that the towers had collapsed.

Josh: What's in store for the next 20 years of Cutting Edge?

Dino: AI in some way, I guess, but not sure which way yet. Since 1998 we've seen the advent of the Internet as a first-class developer platform, the rise of scalability as a problem, and the evolution of serious SQL and then NoSQL databases. We saw the browser evolve from a target for simple JavaScript development into something more. Developers began writing browser-based Web apps in Silverlight, then in C#, and then in JavaScript again. On the data front we went from plain data access to object/relational mapping (ORM) and now toward micro ORM.

Those who start with computers today should be aware that programming languages are like any tool. Handling it well helps, but you can only do a good job if you know how and where to use it. ■

Dino Esposito *is the author of "Microsoft .NET: Architecting Applications for the Enterprise" (Microsoft Press, 2014) and "Programming ASP.NET Core" (Microsoft Press, 2018). Pluralsight author and developer advocate at JetBrains, Esposito shares his vision of software on Twitter: @despos.*

Visual Studio LIVE!

EXPERT SOLUTIONS FOR .NET DEVELOPERS

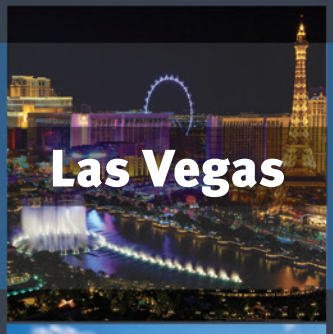
Visual Studio LIVE! 25 YEARS OF CODING INNOVATION

VISUAL STUDIO LIVE! (VSLive!™) is celebrating 25 years as one of the most respected, longest-standing, independent developer conferences, and we want you to be a part of it. Join us in 2018 for #VSLive25, as we highlight how far technology has come in 25 years, while looking toward the future with our unique brand of training on .NET, the Microsoft Platform and open source technologies in seven great cities across the US.

HELP US CELEBRATE #VSLIVE25. WHICH LOCATION WILL YOU ATTEND IN 2018?

MARCH 11 – 16
Bally's Hotel & Casino

Respect the Past. Code the Future.



APRIL 30 – MAY 4
Hyatt Regency Austin

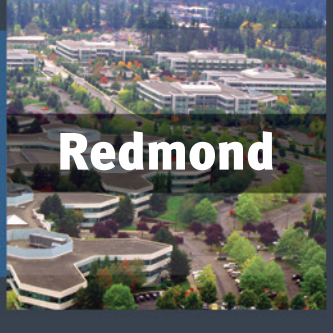
Code Like It's 2018!

JUNE 10 – 14
Hyatt Regency Cambridge

Developing Perspective.

AUGUST 13 – 17
Microsoft Headquarters

Yesterday's Knowledge; Tomorrow's Code!



SEPTEMBER 17 – 20
Renaissance Chicago

Look Back to Code Forward.

OCTOBER 7 – 11
Hilton San Diego Resort

Code Again for the First Time!

DECEMBER 2 – 7
Loews Royal Pacific Resort

Code Odyssey.

CONNECT WITH US

twitter.com/vslive – @VSLive

facebook.com – Search "VSLive"

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

SUPPORTED BY

Microsoft Visual Studio msdn magazine Visual Studio MAGAZINE

PRODUCED BY

1105 MEDIA YOUR GROWTH. OUR BUSINESS.

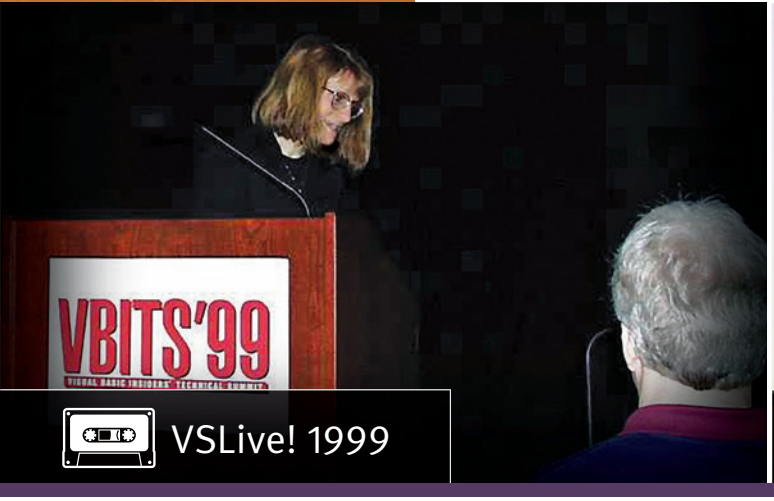
vslive.com

#VSLIVE25

We're Gonna Code Like It's 2018

Visual Studio Live! (VSLive!™) is back in Austin this spring with 5 days of practical, unbiased, Developer training. From April 30 – May 4, 2018, join us for hard-hitting sessions, insightful workshops, intense hands-on labs and fun networking events. Code with industry experts, hear the latest from Microsoft insiders and tune up on today's hottest training topics! Plus, help us celebrate 25 years of coding innovation as we take a fun look back at technology and training since 1993. Come experience the education, knowledge-share and networking at #VSLive25.

Visual Studio LIVE! 25 YEARS OF CODING INNOVATION



SUPPORTED BY

Microsoft · Visual Studio · msdn magazine · Visual Studio MAGAZINE

PRODUCED BY

1105 MEDIA
YOUR GROWTH. OUR BUSINESS.

DEVELOPMENT TOPICS INCLUDE:

- VS2017/.NET
- JavaScript / Angular
- Xamarin
- Software Practices
- Database & Analytics
- ASP.NET / Web Server
- ALM / DevOps
- Azure / Cloud
- UWP (Windows)
- Hands-On Labs — BACK BY POPULAR DEMAND

"I have been looking forward to attending the event for some years and I was finally able to come. I need to help keep my company current and secure. I am advocating for more test-driven dev and using more async as a resulting of attending the conference!"

– Yessenia Figueroa, 1-800-Contacts

"This is my second year attending Visual Studio Live! I want to come back year over year because the variety of sessions allow me to deep dive into tech that I use on a daily basis, but the option is there to explore something new. The biggest dev improvement I made last year was from the great hands-on experience I got with SignalR at VSLive!"

– Jake Clauson, Washington Technology Solutions

Register to code with us today!

Register Now and Save $300!

Use promo code VSLJAN2

CONNECT WITH US

twitter.com/vslive – @VSLive

facebook.com – Search "VSLive"

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

vslive.com/austinmsdn

Developing Perspective

Visual Studio Live! (VSLive!™) returns to Boston, June 10 – 14, 2018, with 5 days of practical, unbiased, Developer training, including NEW intense hands-on labs. Join us as we dig into the latest features of Visual Studio 2017, ASP.NET Core, Angular, Xamarin, UWP and more. Code with industry experts AND Microsoft insiders while developing perspective on the Microsoft platform. Plus, help us celebrate 25 years of coding innovation as we take a fun look back at technology and training since 1993. Come experience the education, knowledge-share and networking at #VSLive25.

Visual Studio LIVE! 25 YEARS OF CODING INNOVATION

SUPPORTED BY

Microsoft Visual Studio msdn magazine Visual Studio MAGAZINE

PRODUCED BY

1105 MEDIA YOUR GROWTH. OUR BUSINESS.

DEVELOPMENT TOPICS INCLUDE:

- VS2017/.NET
- JavaScript / Angular
- Xamarin
- Software Practices
- Database & Analytics
- ASP.NET / Web Server
- ALM / DevOps
- Azure / Cloud
- UWP (Windows)
- Hands-On Labs NEW!

"I have attended 12+ Visual Studio Live events, starting with the early days of VBITS and C# Live. I return every year because no one reflects and enhances what is currently happening in and with Microsoft's developer world like VSLive! I honestly trust and enjoy all the speakers I look to them as both mentors and old friends!"

– John Kasarda, Accuquote

"Last year, I got so much out of the SOLID principle sessions – it gave me great perspective on the subject. This year, I loved the HoloLens introduction. I work for an aerospace company, and I can see that the technology will be utilized for assembly, 3D design, 3D inspection and many other applications in the near future!"

– Mani Nazari, IT Factory Automation Systems

Register to code with us today!

Register Now and Save $300!

Use promo code VSLJAN2

CONNECT WITH US

twitter.com/vslive – @VSLive

facebook.com – Search "VSLive"

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

vslive.com/bostonmsdn

WD-40

In my November 2017 column about duct tape (msdn.com/magazine/mt845623), I promised you a follow-up piece on duct tape's polar opposite. Herewith my ruminations on the all-purpose penetrating lubricant WD-40 (wd40.com), and its application to software.

As duct tape can stick almost anything together, WD-40 can get almost anything apart. You'll find them nestled next to each other in almost every toolbox in the world. This confluence is celebrated in song, see bit.ly/2hW4IOy and bit.ly/2BjLvhz.

Our applications use duct tape internally, where the user can't see it—lots and lots of duct tape. But the external functionality of our most successful software owes more to WD-40. Our apps rarely invent something completely new. What we usually do is reduce the friction of existing operations, as WD-40 reduces the friction of mechanical parts.

Consider Uber, the much-maligned but much-much-used ride hailing service. Uber isn't doing anything conceptually new. New York City has had radio-dispatched car services for as long as it's had radios and cars: call 777-7777 and you get Dial 7's car service; call 666-6666 and you get Carmel's. The same business structure applies: drivers own their cars and do the driving; the central service handles marketing, booking and payment. Similar operations exist in many other cities.

How has Uber reduced the friction of this process? With Uber, you summon a ride without talking to a live person. The younger generation especially prefers this, and it's one less body Uber needs to hire; lowering costs permanently once you get the code written. The app remembers your most frequent destinations, offering them at a single tap. You don't need to find a new car service in each city you visit, the same app works almost everywhere. You don't have to describe or even know your location in a strange city, the app takes care of that. You don't have to visit an ATM for cash, or even take out your credit card; the service handles all of that. The driver doesn't need an expensive radio console to participate, just the smartphone she probably already carries. A good squirt of WD-40 on all the joints limbers up a service that is pulverizing the competition.

Now consider Amazon. It's not doing anything conceptually that Sears didn't do with their mail-order operation a hundred years ago, but Amazon is drastically lowering friction and hammering Sears. Amazon's Web site is easier to distribute and update than the Sears paper catalog. Items are easier to find with a search box than a paper index. Amazon's 1-Click Order button is much easier than filling out and snailing a paper order form with a paper check. And customers no longer have to wait a week or longer for the Wells Fargo Wagon to deliver their orders, holding a town parade when it arrives, as they did in "The Music Man" (straight version at bit.ly/2AaVng8, spoof at bit.ly/2ACLSYa).

Amazon has dumped a barrel of WD-40 into all aspects of the retail process, lowering the friction nearly to zero. I had to turn 1-Click off because I was buying too much stuff. Amazon founder Jeff Bezos became the world's richest man in late October. "The Music Man" song will soon need rewriting: "Oh, the Amazon drone is a-flying down the street …"

> As duct tape can stick almost anything together, WD-40 can get almost anything apart.

We should use more WD-40 in designing software. Maybe, in addition to posters displaying our target user personas, we should display big spray cans of WD-40 around the lab as an inspiration to ask ourselves, "How can we lower friction even more?" For example, the auto-save in One Note has lower friction than manual saving in Word. The automatic disk backup of Carbonite has much lower friction than backing up data yourself. Amazon's patent on 1-Click ordering expired last fall (see bit.ly/2jq9YdC), so I expect to see many companies start using it. Sears, on the other hand, brought back their paper catalog this Christmas (cnnmon.ie/2zCHS62), betting that nostalgia could bring back buyers. I'm betting it was the last Christmas for Sears as a going concern.

The night before code freeze, we shouldn't be trying to cram in one more feature for our few power users. We should be trying to lower the friction for everyone. We should be asking ourselves, "Where else in this process can we spray WD-40?" ■

DAVID S. PLATT *teaches programming .NET at Harvard University Extension School and at companies all over the world. He's the author of 11 programming books, including "Why Software Sucks" (Addison-Wesley Professional, 2006) and "Introducing Microsoft .NET" (Microsoft Press, 2002). Microsoft named him a Software Legend in 2002. He wonders whether he should tape down two of his daughter's fingers so she learns how to count in octal. You can contact him at rollthunder.com.*

GrapeCity | Developer Solutions

Empower your development. Build **better** applications.

GrapeCity's family of products provides **developers**, **designers**, and **architects** with the ultimate collection of easy-to-use tools for building **sleek**, **high-performing**, **feature-complete** applications. With over 25 years of experience, we understand your needs and offer the industry's best support. **Our team is your team.**

Visual Studio Magazine 2 READER'S 2017 CHOICE

Visual Studio Magazine 3 READER'S 2017 CHOICE

Call **1-800-831-9006**
Use code MSDN0118 to get **10% off**

* Offer applies to products in ComponentOne, ActiveReports, Spread, and Wijmo lines. To take advantage of the offer, the promo code should be applied in the cart at checkout. Offer excludes ActiveReports Server, distribution licenses, volume discount packs, and add-on support/maintenance. Not to be combined with other offers. Other restrictions may apply.

ComponentOne
.NET UI CONTROLS

ActiveReports
REPORTING SOLUTIONS

Spread
SPREADSHEET SOLUTIONS

Wijmo
JAVASCRIPT UI CONTROLS

For more information: **1-800-831-9006**

Learn more and get free 30-day trials at **GrapeCity.com**

Visual Studio Microsoft Partner

FULL SUPPORT FOR Visual Studio 2017