THE MICROSOFT JOURNAL FOR DEVELOPERS

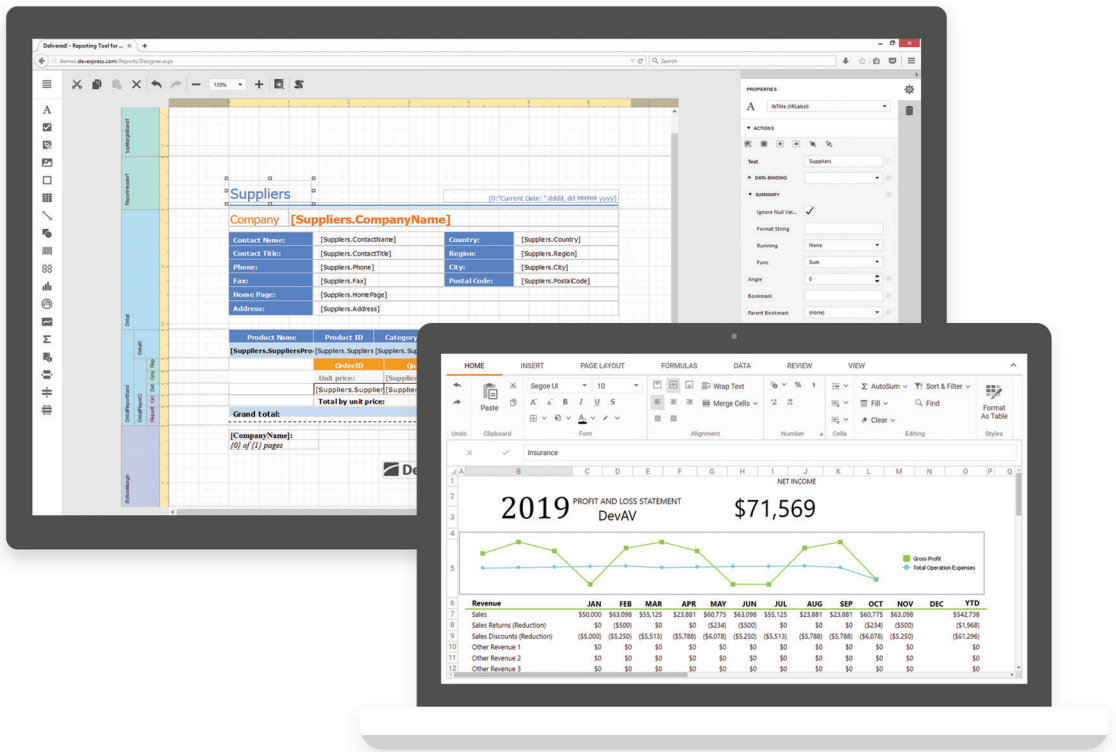
NOVEMBER 2019 VOL 34 NO 11

msdn magazine

DevExpress®

DevExpress ASP.NET Subscription

Create high performance line-of-business applications that meet and exceed the needs of your enterprise.



ASP MVC .NET CORE B JS BLZ

The DevExpress ASP.NET Subscription includes over 200 components and business users across platforms and devices is a simple matter of selecting the appropriate UI elemen and dropping data fields onto corresponding arguments, values, and series. And because DevExpress Dashboard automatically provides the best data visualization option for you, results are immediate, accurate and always relevant.

To learn more, download our free 30-day trial today.

devexpress.com/try

THE MICROSOFT JOURNAL FOR DEVELOPERS

NOVEMBER 2019 VOL 34 NO 11

msdn magazine

The Impact of Exponential Technologies.....28

COLUMNS

Microsoft

INFRAGISTICS

Faster Paths to Amazing Experiences

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

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

Get started today with a free trial:
Infragistics.com/Ultimate

To speak with sales or request a product demo with a solutions consultant call 1.800.231.8588

Delivering amazing experiences for 30 years.

Infragistics Ultimate

- Fastest **grids** & **charts** on the market – any device, any platform
- Build Spreadsheets with Charts in WPF, Windows Forms, Angular & JavaScript
- Get Angular code from Sketch designs with Indigo.Design
- 100% support for .NET Core 3

msdn magazine

NOVEMBER 2019 VOLUME 34 NUMBER 11

ID STATEMENT *MSDN Magazine* (ISSN 1528-4859) is published 13 times a year, monthly with a special issue in December by 1105 Media, Inc., 6300 Canoga Avenue, Suite 1150, Woodland Hills, CA 91367. Periodicals postage paid at Woodland Hills, CA 91367 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*, File 2272, 1801 W.Olympic Blvd., Pasadena, CA 91199-2272, email MSDNmag@1105service.com or call 866-293-3194 or 847-513-6011 for U.S. & Canada; 00-1-847-513-6011 for International, fax 847-763-9564. POSTMASTER: Send address changes to MSDN Magazine, P.O. Box 2166, Skokie, IL 60076. Canada Publications Mail Agreement No: 40612608. Return Undeliverable Canadian Addresses to Circulation Dept. or XPO Returns: P.O. Box 201, Richmond Hill, ON L4B 4R5, Canada.

CORPORATE ADDRESS 1105 Media, Inc.
6300 Canoga Avenue, Suite 1150, Woodland Hills 91367
1105media.com

MEDIA KITS Direct your Media Kit requests to Chief Revenue Officer Dan LaBianca, 972-687-6702 (phone), 972-687-6799 (fax), dlabianca@Converge360.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
Email: 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
2121 Alton Pkwy., Suite 240, Irvine, CA 92606

Corporate Office (weekdays, 8:30 a.m.-5:30 p.m. PT)
Telephone 818-814-5200; Fax 818-734-1522
6300 Canoga Ave., Suite 1150, Woodland Hills, CA 91367

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

Microsoft

BPA WORLDWIDE

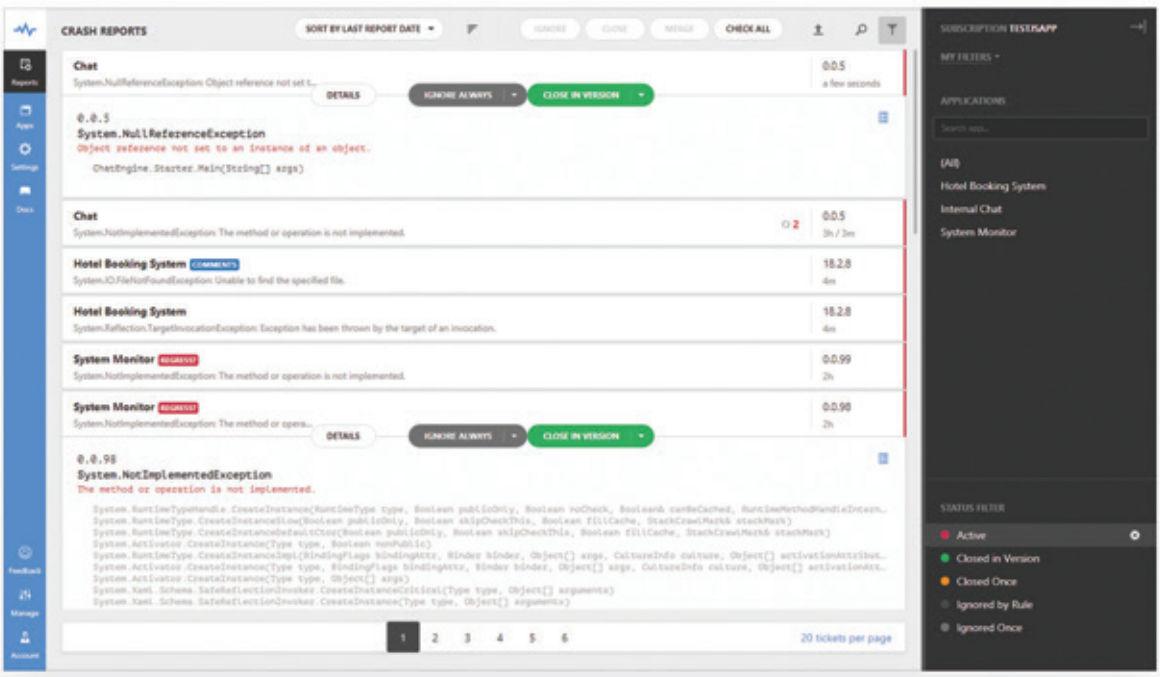
1906 100 2006 AMERICAN BUSINESS MEDIA

WPA Western Publications Association

logify™

Detect Errors and Exceptions
with 24/7 Application Monitoring

Logify's cloud-based application monitoring service allows you to automatically collect app crash events and runtime exceptions. With Logify, you will never have to deal with Visual Studio's® internal exception information. Logify presents all relevant exception data in an easy-to-understand, clutter-free manner. From loaded modules and cookies to browser info, OS build, user activity — Logify will organize results so you can address application issues in the shortest possible time.



Learn More Today — Try Logify Risk Free for 15 days

devexpress.com/logify

DevExpress®

Editor's Note

MICHAEL DESMOND

Hail and Farewell

I was named after my grandfather, a newspaper man out of Toledo, Ohio, who died of congestive heart failure at the age of 65, soon after I was born. He never got to meet me, but my parents would tell the story of the note he sent me from his hospital bed. It read simply: "Ave atque vale." Hail and farewell.

It is a time of farewells here at *MSDN Magazine*, as we close the final issue in the publication's 33-year history. I've been editor in chief of this magazine for eight years, nearly one-quarter of its total run, and have had the opportunity to forge so many connections and friendships. I've enjoyed the subversive humor of David Platt and Ted Neward, experienced the unpredictable genius of Thomas Hansen, and marveled at the ridiculous work ethic of Stefano Tempesta.

And that's the point, really. *MSDN Magazine* and its decades of success wasn't about the code, as much as it was about the people. Our long-running stable of columnists—Julie Lerman, James McCaffrey, Ted Neward, David Platt, Frank La Vigne and Dino Esposito—is proof of that, as are frequent contributors like Mark Michaelis, Alessandro Del Sole, Steve Smith and Srikantan Sankaran. Rachel Appel kick-started our exploration of modern apps, Jonathan Waldman produced some of the best blockchain articles I've ever read, and Bruno Terkaly and Ricardo Villalobos tackled Azure back when it was a thing developers regarded with suspicion.

The magazine has seen so many brilliant contributions from the field, from authors like Vassili Kaplan, Hadi Brais, Laurent Bugnion and Zvi Topol. And the list of Microsoft contributors goes on and on. Stephen Toub, Kraig Brockschmidt, Ashish Sahu, Dmitry Lyalin, Stephen Kerr, Adam Tuliper, Kevin Ashley, Andrew Whitechapel, Immo Landwerth. To all of you who took time from your demanding jobs to help inspire and enlighten our readers, thank you.

I want to recognize also those who came before. To former editors in chief Keith Ward, Howard Dierking, Stephen Toub, Michael Longacre, Eric Maffei and John Lazarus (who started it all), we stood on the shoulders of giants. And thank you to our editorial directors at Microsoft, who at times served as defender, advocate, liaison and muse. Diego Dagum, Kit George, Jennifer Mashkowski and Mohammad Al-Sabt—your efforts enabled us to do some great things.

I won't neglect our amazing staff. To Features Editor Sharon Terdeman, whose skill, dedication and patience transformed even the most calamitous manuscripts into polished, finished articles; and to Art Director Josh Gould, who brought visual clarity and cohesion to our uniquely challenging content, thank you! And a shout out to former technical editors David Ramel and Terry Dorsey, who displayed a rare blend of developer expertise and editing skill. Finally, Wendy Hernandez, our tireless managing editor, you made all of us look good. You were the engine at the heart of *MSDN Magazine*, and its success is a reflection of your ability to keep me from wandering off—because you know I'll wander off.

Well, that's it then. It's time to close the book on *MSDN Magazine*. But know this: The people who breathed life into the magazine are still out there, testing the edges of new frameworks, exploring the limits and possibilities of the tools and languages, and sharing their insights with the community. The body of this enterprise may be at an end, but the spirit of it will live on.

Ave atque vale, my friends.
Hail. And farewell.

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

File Format APIs

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

Try risk free - 30 day trial

ASPOSE

Download a Free Trial at

https://downloads.aspose.com

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

Deliver efficient applications to Create, Draw, Manipulate or Convert image file formats.

Aspose.Tasks

Develop high performance apps to Create, Edit or Convert Microsoft Project® document formats.

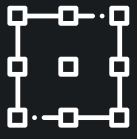
Aspose.Diagram | Aspose.Note | Aspose.3D | Aspose.CAD | Aspose.HTML | Aspose.GIS

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

NEO Blockchain

The F5 Developer Experie

Smart Contract
Debugger



NEO
Express

Smart Co
Compiler Enh

Get started today

neo.org/dev

github.com/neo-project/neo-blockchain-toolkit

Toolkit for .NET

nce for Smart Contracts

ntract
ancement

Visual
DevTracker

NEO-FX
Library

JULIE LERMAN

Backing Field and Owned Entity Changes in EF Core 3.0

Well, this is it—the final issue of *MSDN Magazine*, and the final iteration of my column. I've decided to write about two of my favorite topics, Entity Framework Core (EF Core) and Domain-Driven Design (DDD). I've explored using Entity Framework (EF) and EF Core as a mapper between DDD-influenced domain models and relational databases a number of times over the years. Now EF Core 3.0 has just been released (in September 2019), so I'll wrap up with a look at some of the improvements that help with these mappings.

My most recent exploration was expressed in a three-part series about EF Core 2.0 as a mapper between DDD models and relational databases. I'll focus in these last pages on changes brought with EF Core 3.0 that improve the experiences of encapsulating logic with backing fields and of mapping value objects.

Backing Basics in EF Core

EF Core introduced support for backing fields, and each iteration brought a better experience with increased support. While backing fields aren't specific to DDD, they do allow you to encapsulate properties of your classes, and that's very important for DDD. The encapsulation lets you more easily control interaction with your classes and APIs to ensure they aren't intentionally or accidentally misused.

EF6 only maps to properties, not to fields, though it does allow you to protect the value of a property by using a private setter. Because EF uses reflection to materialize query results, it is able to set the value of the property. But the property itself needed to be public:

```
public string Name {get; private set;}
```

That covered a lot of cases, but not if you really wanted the property to be private. A common use case might be the key property. EF needs to interact with it, but you may not want developers to read it directly or even use it in queries.

EF Core's backing field support can map to backing fields whether you explicitly define backing fields or rely on inferred backing fields to comprehend properties. This allows EF Core to recognize even private properties when creating migrations, as well as to populate them when materializing query results. In this case I'm not even bothering with a getter and setter, but just using an expression body to return _companyId:

```
private int _companyId;
private int CompanyId =>_companyId;
```

Code download available at msdn.com/magazine/1119magcode.

If the property is private, but you don't declare a backing field, EF Core will infer it for you when building the data model. In the specific case of making the key private, the ModelBuilder won't be able to see the private property, so you'll need to tell EF Core that it's the key:

```
modelBuilder.Entity<Company>().HasKey("CompanyId");
```

EF Core is able to see that there's a backing field (_companyId) related to the property so it can materialize results to that field, which will then feed the property. If the backing field doesn't align with the property by convention, you can pair them up with a configuration in the DbContext.

Up through 2.2, EF Core would default to reading and writing property values through an available getter and setter even if there was a backing field available, with one exception—when materializing objects from queries. This exception ensured that any special business logic in a setter would not be triggered just because you were retrieving data from the database.

For example, if you edit the EmployeeCount of the company class, the business rule has a message fired off about the change:

```
private int _employeeCount;
  public int EmployeeCount
  {
    get { return _employeeCount; }
    private set
    {
      _employeeCount = value;
      SendEmployeeCountChangedMessage();
    }
  }
```

That doesn't make sense if you're just retrieving the company data from the database. So, because there's a backing field available, EF Core would set the value using the _employeeCount field.

Backing Field Changes in EF Core 3.0

The team wanted to make sure that as a mapper, EF Core 3.0 aligns better with how we think about object persistence rather than just tying everything to public properties. From Microsoft Program Manager Diego Vega:

"EF Core's goal should be to facilitate persisting and rehydrating objects from the database, and that is better achieved by mapping the object's state, which is all represented by the object's fields (even for automatic properties, there is always a backing field). On the other hand, the decision to add a public property on an object has more to do with the intent to expose specific data from the object to the rest of the program, and should not be determined by persistence concerns." This truly brings us back to our DDD-focused design.

LEADTOOLS®

Integrate LEADTOOLS OCR, powered by state of the art machine learning algorithms

- Fast, accurate, and reliable optical character recognition for use in any application or environment
- Utilize multiple cores for unparalleled performance
- Supports multiple text recognition engines including OCR, ICR, MICR, MRZ, and MRP
- Automatically detect zones, paragraphs, and recognize multiple languages on the same document
- Output text searchable document formats such as PDF, PDF/A, DOC, DOCX, XML, XPS, and more

Microsoft .NET · Xamarin · Windows · Java · Linux · iOS · macOS · Android · HTML5

Get Started Today

DOWNLOAD OUR FREE EVALUATION

LEADTOOLS.COM

Vega continues, "There are always going to be cases in which you need to exclude some field (or property) from the mapping, because the state in the field represents some calculated value that is cached in memory but doesn't need to be persisted, but mapping to backing fields provides a superior default starting point than mapping to public properties."

So the biggest change to backing fields in EF Core 3.0 is that EF will always map to backing fields in all cases rather than only when querying.

Even with the exception of queries, there had been a case reported where the query wasn't using the backing field—when the backing field was a navigation property. Here's an example where an employee class has a navigation property to the employee's company. If I edit the employee's company using the navigation property, that should trigger some other behavior, in this case sending a message, like so:

```
private Company _company;
public Company Company
{
  get { return _company; }
  set
  {
    _company = value;
    SendEmployeeChangedCompany();
  }
}
```

Again, it doesn't make sense for that to happen if I'm just populating the Company property using a query such as:

```
context.Employees.Include(e=>e.Company).FirstOrDefault();
```

With EF Core 3.0, rather than specifying this additional case for querying, the broader decision (explained earlier) was made to always read or write to a backing field if it exists.

That's the new default. However, remember (or note if this is new to you) that you can fine-tune how a DbContext uses backing fields for properties in the ModelBuilder. Its UserPropertyAccessMode setting can be set to one of six enums: Field (the default), FieldDuringConstruction, PreferField, PreferFieldDuringConstruction, PreferProperty and Property.

Because this is a breaking change, you can set the value back to the old default (PreferFieldDuringConstruction) or use your tests to see where the behavior is changing in order to align your code.

The other two changes to backing fields cover narrow cases. The first issue occurred because the naming convention for matching fields with properties only checked the name and had an order of precedence:

1. _<camel-cased property name>
2. _<property name>
3. m_<camel-cased property name>
4. m_<property name>

It was possible to match a field to the wrong property—possibly even a property of the wrong type—if more than one field fit the pattern. Here's an exaggerated example:

```
private string _name;
public string Name => _name;
private string m_name;
public string OwnerName => m_name;
```

Hopefully, your DDD practices and use of a ubiquitous language will ensure you never find yourself in this situation. However, in EF Core 3.0, if more than one backing field matches a property, it will throw a runtime exception, rather than quietly making what could be the wrong choice.

The real problem with the preceding code is that OwnerName and m_name don't fit any convention and you have to map them together regardless. This will leave only one backing field choice for Name, and EF Core will be happy:

```
modelBuilder.Entity<Company>().Property(c=>c.OwnerName).HasField("m_name");
```

But if you neglect to map OwnerName to m_name, EF Core 3.0 will report the ambiguous naming issue.

The last change is for backing fields that don't map to any property yet still need to be persisted. Imagine you have a field, _favoriteEmployee, in your class but no property. Yet you need that field value to get persisted for a reporting system to access:

```
private string _favoriteEmployee;
public void SetFave(string fave)
{
  _favoriteEmployee=fave;
}
```

The model still needs to associate the field with a property in order for this to be persisted. You can achieve this by configuring a property and mapping it with HasField to the backing field. For example:

```
modelBuilder.Entity<Company>()
  .Property<string>("FavoriteEmployee").HasField("_favoriteEmployee");
```

Interestingly, because you map inferred properties in the same manner as shadow properties, I mistakenly referred to them as shadow properties at first. But shadow properties are for data that's only in the data store and has no representation—not even a field—in the class.

To avoid issues with ambiguity that developers had been encountering and to allow for cleaner API code, starting with EF Core 3.0, the shadow property name must match the field name exactly. Otherwise it will throw a runtime exception:

```
modelBuilder.Entity<Company>()
.Property<string>("_favoriteEmployee").HasField("_favoriteEmployee");
```

In a follow-up release, you'll be able to configure field names that don't match exactly.

Using backing fields to encapsulate properties also extends to navigation properties and properties that are mapped as owned entities (commonly used for value objects).

EF Core enables encapsulation of collections, thanks to it being able to comprehend IEnumerable properties. This was a big deal for DDD practitioners because there was no way to do this in EF6 or earlier versions. I wrote about it in an earlier column, EF Core 1.1: A Few of My Favorite Things (msdn.com/magazine/mt745093). Nothing has changed with EF Core 3.0 in this regard.

Important Changes to Owned Entities in EF Core 3.0

Owned entities (also called owned types) are the feature that allows you to map value objects in EF Core. Owned entities in EF Core were the result of re-thinking the complex type feature that's been in EF since the beginning. The recipe for owned types in EF Core is as follows:

- The type has no key property of its own.
- The type is used as a property of one or more entities (or even another owned entity).
- The DbContext contains an OwnsOne or OwnsMany mapping that defines the property as an owned type for each entity where it's used as a property.

Address the ELEPHANT IN THE ROOM

Bad address data costs you money, customers and insight.

Melissa's 30+ years of domain experience in address management, patented fuzzy matching and multi-sourced reference datasets power the global data quality tools you need to keep addresses clean, correct and current. The result? Trusted information that improves customer communication, fraud prevention, predictive analytics, and the bottom line.

- Global Address Verification
- Digital Identity Verification
- Email & Phone Verification
- Location Intelligence
- Single Customer View

+ .NET
+ Microsoft® SSIS
+ Microsoft® Dynamics CRM

See the Elephant in Your Business -

Name it and Tame it!

melissa

www.Melissa.com | 1-800-MELISSA

Free API Trials, Data Quality Audit & Professional Services.

I have a PersonFullName type with two string properties (First and Last) and no key property. This type also has other attributes of a value object. You can see its code in the accompanying download. And you'll find a short primer on value objects in my column at msdn.com/magazine/mt846463.

The PersonFullName value object is used as a property of Employee:

```
public PersonFullName Name { get; private set; }
```

For EF Core to persist the value object along with the employee, I need to provide an explicit mapping in the DbContext:

```
modelBuilder.Entity<Employee>().OwnsOne(e=>e.Name);
```

All of this was available beginning with EF Core 2.0. A minor but handy fix that arrived in a later version of EF Core is related to constructors. EF Core can instantiate entities if there's a constructor whose parameters match the properties of the entity. But owned entities didn't allow this. However, that's now been fixed, which is especially nice for value objects, as they need their property values at instantiation to avoid creating objects in an invalid state.

EF Core 3.0 Fix for a Significant Owned Entities Problem There was a big flaw in the original behavior of owned entities that has now been fixed in EF Core 3.0. By default, EF Core maps the values of the owned entity to columns in the same table where the owner data is stored. But EF Core required that an owned property (for example, Employee.Name) be populated so that it could read its values (such as First and Last). This meant you could never leave the Name property null even if your business rules allowed it. With the help of some of the EF team members, I came up with a work around that I wrote about in the previously mentioned article.

This problem was not only tied to owned entities. It also affected related entities where the mappings forced a dependent entity's data to be stored in the principal entity's table.

A change in EF Core 3.0 fixes this problem. Entities that are owned or are dependents in a one-to-one relationship are now optional. In the case of my PersonFullName value object, if I don't yet know the name of the new employee (for some strange reason), I can still create and persist an employee object.

Another issue in an earlier version of EF Core that has been corrected was that EF Core didn't support updating or replacing owned entity properties. But this is a critical pattern to support because value objects are immutable and the owned entity mapping is used for value objects, so the only way to "edit" the owned property is to replace it with a new instance. The issue was fixed in a prior update to EF Core and now the following code (which retrieves an entity, replaces its owned Name property, then saves) will persist correctly:

```
var employee = context.Employees.FirstOrDefault();
employee.Name=PersonFullName.Create("Diego", "Vega");
context.SaveChanges();
```

Temporary Workaround for a New Problem There is, unfortunately, a problem arising from the combination of these two fixes: updating owned entities that were originally null. The problem is known (as per this GitHub issue: bit.ly/2maQChH) and is targeted to be fixed in EF Core 3.1.0. In the meantime, if you want to update a null owned entity, Andriy Svyryd from the EF team provides a great workaround (which I've tested in a variety of scenarios) in the referenced GitHub issue. The workaround checks to see if the owner (for example, the employee) is also being added. If so, then Added is the correct state for the owned entity, as well. But if not, then the owned entity can't be new and must therefore have been changed and its state should be marked as Modified.

The workaround takes place in the DbContext SaveChanges override, so your business logic doesn't need to be aware of the temporary fix or that there's even a problem. And when the new fix arrives, you'll only need to remove the extra code in DbContext and won't have to change anything in your business logic. The workaround leverages a newly exposed method called FindOwnership that will return detailed information about an owned entity. I modified the workaround a bit using the IsOwned method and my version is shown in **Figure 1**.

Figure 1 **Override SaveChanges as a Temporary Workaround for a Problem with Replacing Null Owned Entities**

```
public override int SaveChanges () {
  foreach (var entry in ChangeTracker.Entries ()
          .Where (e => e.Metadata.IsOwned () && e.State == EntityState.Added)) {
    var ownership = entry.Metadata.FindOwnership ();
    var parentKey = ownership.Properties
                     .Select (p => entry.Property (p.Name).CurrentValue).
ToArray ();
    var parent = this.Find (ownership.PrincipalEntityType.ClrType, parentKey);
    if (parent != null) {
      var parentEntry = this.Entry (parent);
      if (parentEntry.State != EntityState.Added) {
        entry.State = EntityState.Modified;
      }
    }
  return base.SaveChanges ();
}
```

EF Core Keeps Improving Mappings for DDD

Each iteration of EF Core has brought improvements that enable it to more easily map our nicely designed DDD-influenced domain models. While there haven't been a lot of new features added to EF Core 3.0, a lot of work was done to tighten up the APIs. The team thinks of this as a "foundational" version as it sets the stage for the continued enhancement of EF Core. They're very open to ideas for improvements related to mapping domain models in order to minimize the need to build explicit data models when EF Core's mappings aren't sufficient to the task.

I hope these columns have helped you explore new technologies, expand your knowledge and further your careers! Feel free to reach out to me via my blog at thedatafarm.com. If you want to keep up with me, I send out an occasional newsletter to share articles, upcoming conferences, videos and training opportunities. You can find information about that on my Web site, as well. Thank you for the wonderful opportunity that this column has given me to learn and share with you. ■

Julie Lerman *is a Microsoft Regional Director, Microsoft MVP, Docker Captain and software team coach 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 thedatafarm.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 bit.ly/PS-Julie.*

Thanks to the following Microsoft technical expert for reviewing this article: *Diego Vega*

DocuVieware

Universal HTML5 and Document Management Kit

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
60-day Free Trial Support Included at **www.docuvieware.com**

Powered by **GdPicture**.NET

DocuVieware is an ORPALIS imaging technologies product

Need a software component?

Looking for the best solutions on your platform?

www.componentsource.com

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

Europe (UK)
ComponentSource
The White Building
33 King's Road
Reading, Berkshire
RG1 3AR
United Kingdom

Europe (Ireland)
ComponentSource
Unit 1E & 1F
Core B, Block 71
The Plaza, Park West
Dublin 12
Ireland

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

Call our Licensing Experts:
1.888.850.9911
sales@componentsource.com
Open Mon-Fri (24 hours)

GSA Schedule
Contract GS-35F-0188R

Look no further.

Compare the world's leading component brands at ComponentSource.

DevExpress® GrapeCity® ASPOSE

devart ComponentSpace INFRAGISTICS® HIGHCHARTS Progress® Telerik®

.netCHARTING
/n software
Accusoft
Actipro Software
Active Database Software
ActivePDF
Add-in Express
Advanced Software Eng.
AfterLogic
Allround Automations
Altsoft
amCharts
Antenna House
AnyChart
ArtfulBits
AutoDWG
BALKANGraph
Basic Primitives
BCGSoft
BoostSolutions
CADSoftTools
Catalyst Development Corp.
Catenary Systems
Chant
Clever Components
client IO
Code Effects Software
Codejock Software
combit
Conholdate
Crystal Universe Software
Dart Communications
DBI Technologies
devDept Software
DEVPIA
DidiSoft
DlhSoft
DLSC
Dynamsoft Corporation
EaseFilter
EdrawSoft
Extreme Optimization
Fast Reports
FusionCharts
GdPicture
GEAR.it
GigaSoft
Gridmetric
GroupDocs
Handsoncode
Inlite Research
Intersoft Solutions Corp.
Iocomp Software
Iron Software
jQWidgets
JSCharting
Keyoti
Kinetic Jump Software
Kintivo
KWizCom
Lassalle Technologies
LEADTOOLS
Lidor Systems
Medialooks
MiG InfoCom
MindFusion Group
MultiMedia Soft
Nasosoft Technology
Neodynamic
Nevron
Newtone
Noemax
O2 Solutions
Outside Software
PassMark Software
PDFlib
Perpetuum Software
plexityHide.com
RadiantQ
Rebex
RemObjects Software
Smart HTML Elements
SoftArtisans
Softek Software
Softgroup
Software FX
Steema Software
Stimulsoft
SWIFT Components
Syncfusion
Text Control
Traysoft
TRichView
VectorDraw
VelocityDB
Visual Integrity
Vizuly
WPCubed
Xceed Software
XFINIUM
ZingChart
ZingGrid
ZoomCharts

24/5	20+	950k	180+	30
Customer Service	Years Helping Customers	Licenses Delivered	Countries Served	Day Return Policy

Browse our latest catalog online
www.componentsource.com/catalog

ComponentSource®

TED NEWARD

Python: Functions

Welcome back, Pythonistas. This will be, as many of you know, the last article in this series, but more on that later. In the previous part of this series, I examined how Python flow control works—branching, looping and exception handling—but one of the most significant ways to control (and reuse) code is with the time-honored approach of bundling it into a named block of code that can be invoked repeatedly. I speak, of course, of the traditional function. (Python also supports classes, of course, but much of that should be easy to pick up once you understand the function syntax.)

Predefined Functions

First things first: Unlike many of its object-oriented contemporaries, Python supports (and encourages) top-level functions. In fact, you could program Python entirely in a procedural fashion, if you wanted. (This is partly why Python often gets labeled a "functional" language; more on that later.)

Python has a number of built-in functions defined as part of the core environment, such as the print function I've already discussed. In addition to some core math functions (abs, divmod, max, min, round and pow); some type-conversion functions that transform a variable from one type to another (bool, bin, chr, ord, float, int, hex and oct, among others); and some convenience methods to help construct various types (dict makes it easier to create dictionary instances, and list does the same for lists); Python also has a number of "meta" functions that operate on the language or its elements directly. The eval function, for example, takes a string containing Python and executes that code. Similarly, the compile function takes a string and transforms it into a compiled object for future execution using either eval or its cousin, exec.

> Unlike many of its object-oriented contemporaries, Python supports (and encourages) top-level functions.

But possibly the most "meta" of the Python functions is the dir function, which is named after the shell command of the same name. It returns a list of the names available "inside" the passed object, or the top-level scope if no object is passed. Thus, if you fire up the Python REPL and simply type "dir" into the prompt, Python will respond with all of the global functions and classes that are currently loaded. This is a ridiculously powerful tool that's great to use when exploring new libraries. This reflective vision of the running environment can also be used to examine all of the globals in the module via the globals function, and later, when I start defining some functions, I can examine all of the local variables in the current scope using the locals function.

Function Basics

Defining functions in Python is pretty much exactly the way it is in any other programming language: You define a block of code (properly indented; this is Python, after all) with a name, and the function can take a number of parameters and return a value, like so:

```
def quest():
  print("You must now cut down the tallest tree in the forest...")
  print("With... A HERRING!!!!!")
```

Invoking a function is, as might be assumed from the various print examples used thus far, pretty much exactly as it is in other languages—the name, followed by parentheses enclosing any parameters, and off we go:

```
quest()
```

So far, so good. If you want the function to take parameters, you include one or more names inside the parentheses of the function definition. Any value to be returned is given using the ubiquitous "return" keyword:

```
def theKnightsWhoSayNi(gift):
  if gift == "shrubbery":
    return ("You must now cut down the tallest tree"
      "in the forest... with... A HERRING!!!!")
  else:
    return "You must bring us... A SHRUBBERY!!!"
```

(Notice that when two string literals are adjacent to one another and parenthesized, Python will concatenate them into a single string.) In keeping with Python's dynamically typed nature, any value can be passed for a given parameter, so it's generally a good idea to make sure users know how to use your function correctly; in Python this is done by providing a documentation string ("doc-string") string literal as the first non-commented line of code inside the function, like so:

```
def theKnightsWhoSayNi(gift):
  """In order to continue your quest for Camelot, you must
  bring the Knights gifts. Do not mess with these guys, for
  they say 'Ni' at the slightest provocation!"""
  if gift == "shrubbery":
    return ("You must now cut down the tallest tree"
      "in the forest... with... A HERRING!!!!")
  else:
    return "You must bring us... A SHRUBBERY!!!"
```

DevExpress®

Your Next Great App
Starts Here

x75

When Only the Best Will Do

With the DevExpress Universal Subscription, you'll deliver amazing, high-impact user experiences for Windows®, the web and your mobile world. Get started today and see why your peers consistently vote DevExpress products #1.

Download your free 30-day trial at **devexpress.com** and let's build great apps together.

UI Controls for

DESKTOP / WEB / MOBILE

Unleash the UI Superhero in you and create solutions that fully address customer requirements today and leverage your code investments to build next generation touch-enabled solutions for tomorrow. Whether building an Office® inspired app or a data centric analytics dashboard, DevExpress Universal ships with everything you'll need to build your best, without limits or compromise.

And with industry recognized support services, we'll be with you every step of the way — making certain that our products fulfill their promise. Visit **devexpress.com/support** or email **support@devexpress.com** anytime, with any DevExpress product question.

Geoffrey Jones
Code21 Solutions Pty Ltd

We have been using DevExpress products for over 5 years and been delighted with the results. DevExpress have helped us to develop an enterprise-ready labour management solution with WinForms and ASP.NET modules. As a small team, it would not have been possible for us to develop an application with a polished UI, highly scalable architecture and flexible customization options without the assistance of DevExpress.

Office-Inspired Apps

Touch-Enabled Windows & Web Apps

Native or HTML5 Mobile Apps

Reporting-Dashboard & Analytics Apps

Experience the DevExpress Difference Risk Free

We are so confident in our tools that we back them with a 60-day unconditional money-back guarantee. If in your first 60 days of ownership you find that our products do not meet your business needs, you can return them to us for a full, no questions asked refund.

To get started and to choose a subscription that's right for you, email **info@devexpress.com** or call **+1 (818) 844-3383**.

60 DAY MONEY-BACK GUARANTEE

Grid Controls

Blazing fast, feature-complete and just plain beautiful.

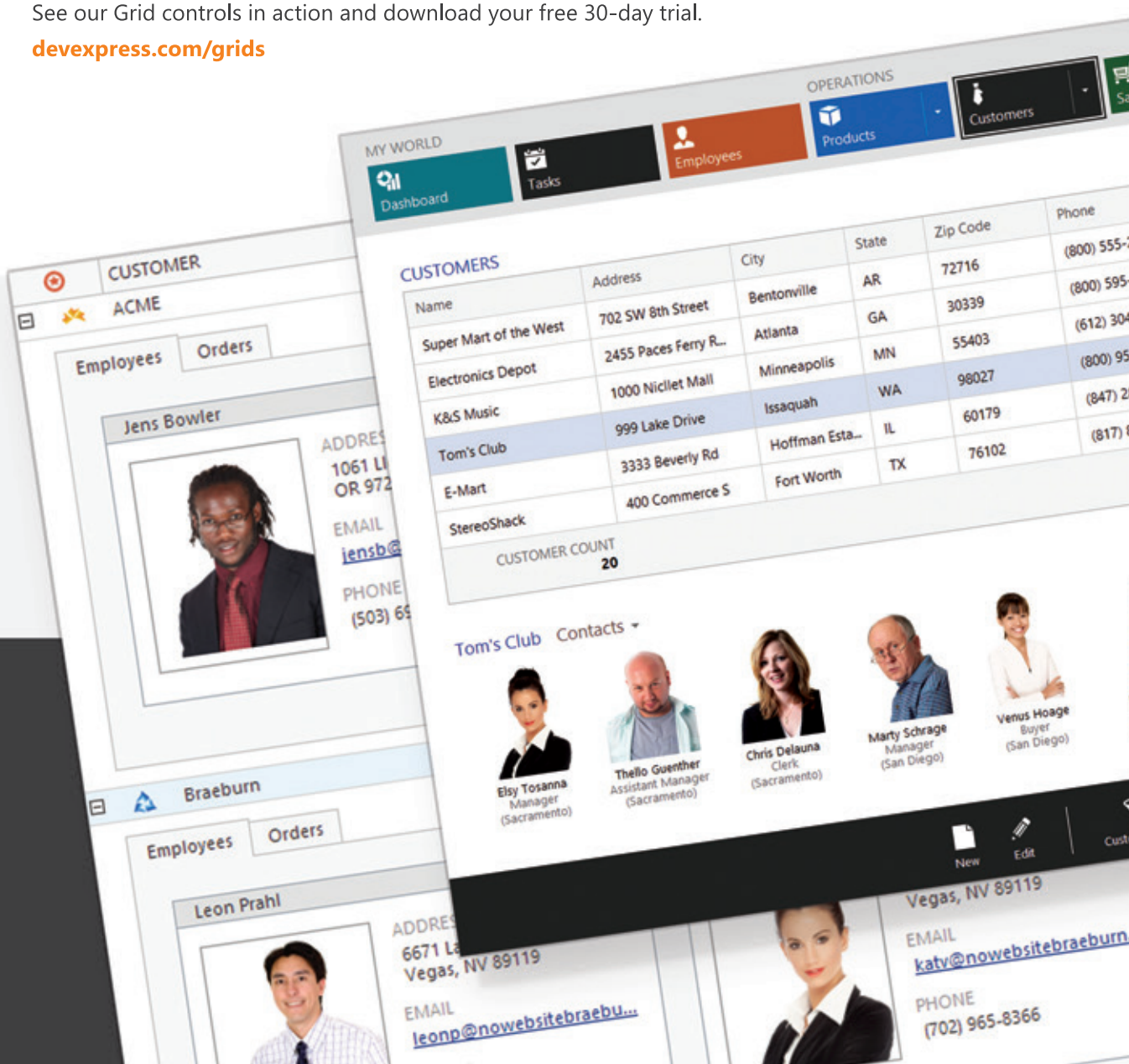
DevExpress Grid controls are Outlook® inspired record editing and data shaping components, allowing your end-users to manage and arrange information on-screen as business requirements dictate. Our data grids ship with dozens of industry leading features including integrated master-detail support and multiple data layout options such as Card and Windows Explorer views. And whether you target desktops or touch devices, all DevExpress Grid Controls support a broad range of user interaction options.

.NET Framework	WinForms / WPF / WebForms / MVC / UWP
.NET Core 3.0	WinForms / WPF / ASP.NET / Blazor
JavaScript	jQuery / Angular / React / Vue

Your Next Great App Starts @DevExpress

See our Grid controls in action and download your free 30-day trial.

devexpress.com/grids



Reporting

Inform and analyze with absolute ease.

The ever changing needs of the enterprise require that reporting platforms offer simple and straightforward end-user customization options so that consumers can freely manipulate report output for maximum clarity. Flexibility is key and ease-of-use paramount.

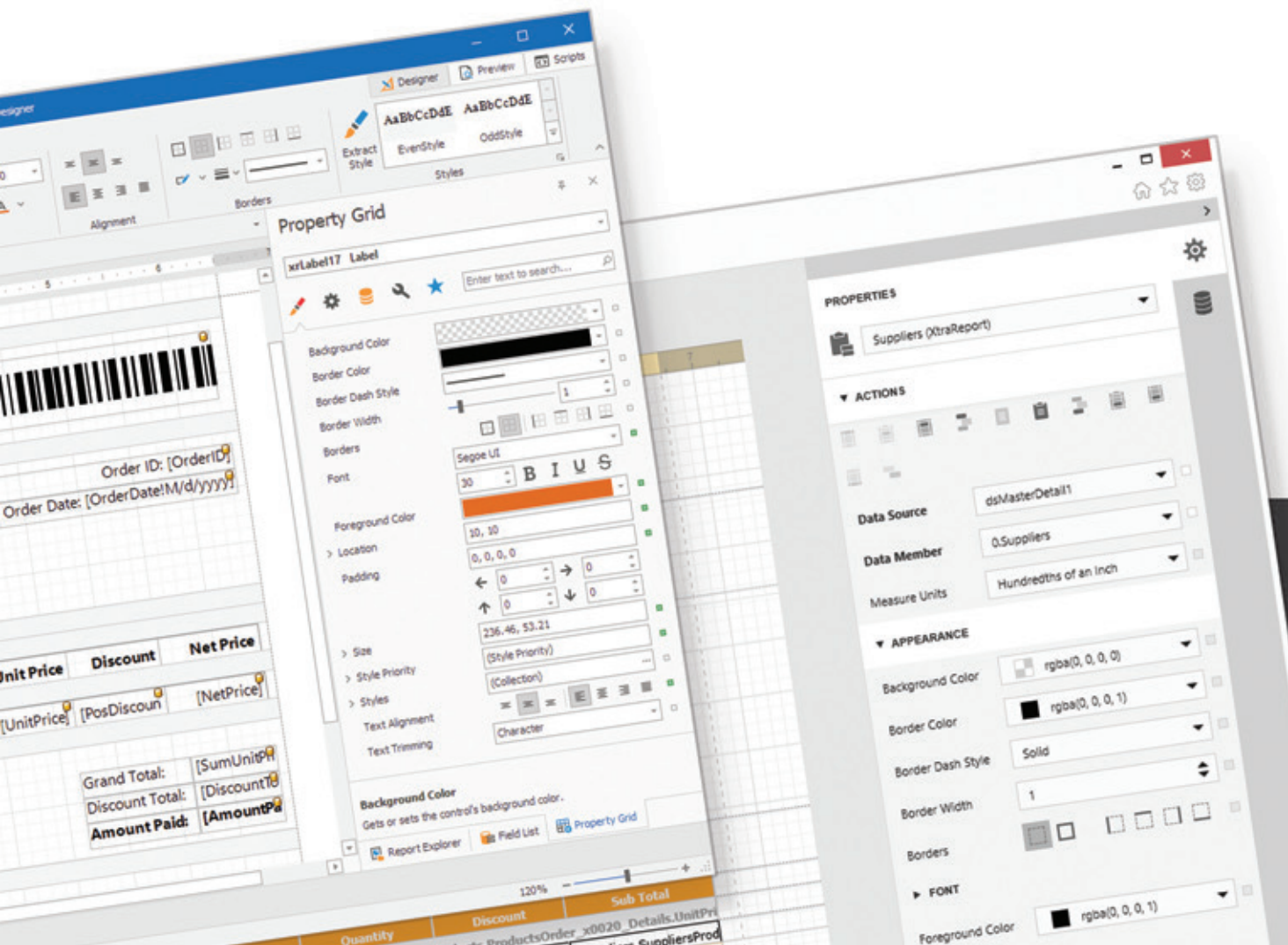
The DevExpress Reporting Suite for .NET was built with your end-users in mind and engineered so you can get the most out of your reporting and data analytics investments. Our Reporting Suite helps you overcome the limitations associated with traditional report tools by providing a fully integrated set of productivity tools, report wizards, pre-built report templates and end-user report designers for both Windows and the web.

.NET Framework	WinForms / WPF / WebForms / MVC
.NET Core 3.0	WinForms / WPF / ASP.NET
JavaScript	jQuery / Angular

Your Next Great Report Starts @DevExpress

See how to solve your Windows® and web reporting requirements forever.

devexpress.com/reports



Spreadsheets (XLS, XLSx, CSV)

The power of Excel® at your fingertips.

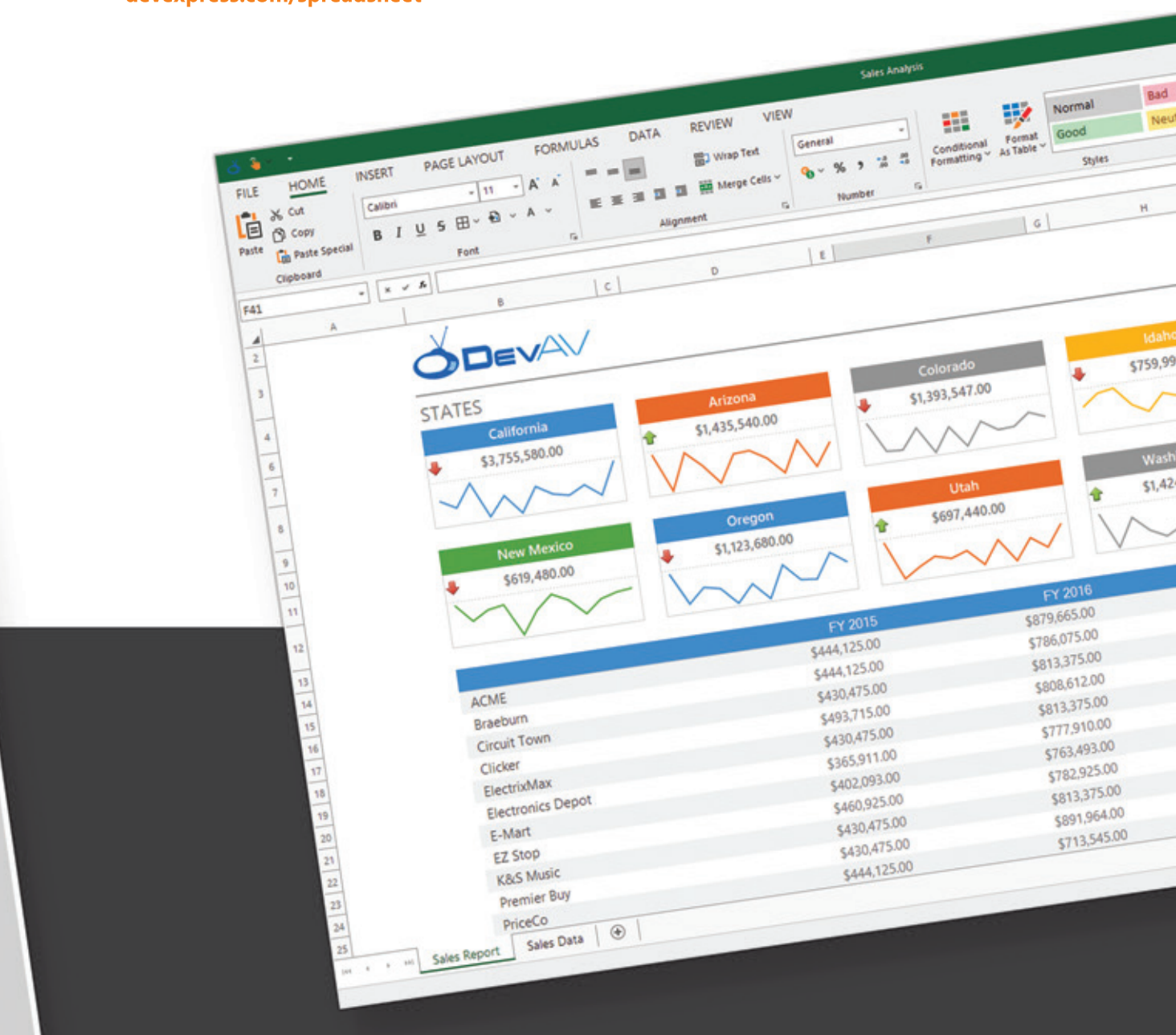
It's no secret...spreadsheets are an important part of business and if you need to incorporate the UX and functionality end-users have come to expect from Microsoft Excel®, DevExpress Spreadsheet is the tool for you. The component library ships with dozens of easy-to-implement features including chart support and fully integrates with other DevExpress components like our Office® inspired Ribbon.

.NET Framework	WinForms / WPF / WebForms / MVC
.NET Core 3.0	WinForms / WPF / ASP.NET

Your Next Great Spreadsheet Starts @DevExpress

See how you can harness the power of spreadsheets in your next .NET app.

devexpress.com/spreadsheet



Word Processing (DOC, DOCx, RTF)

Create the perfect WYSIWYG document or report.

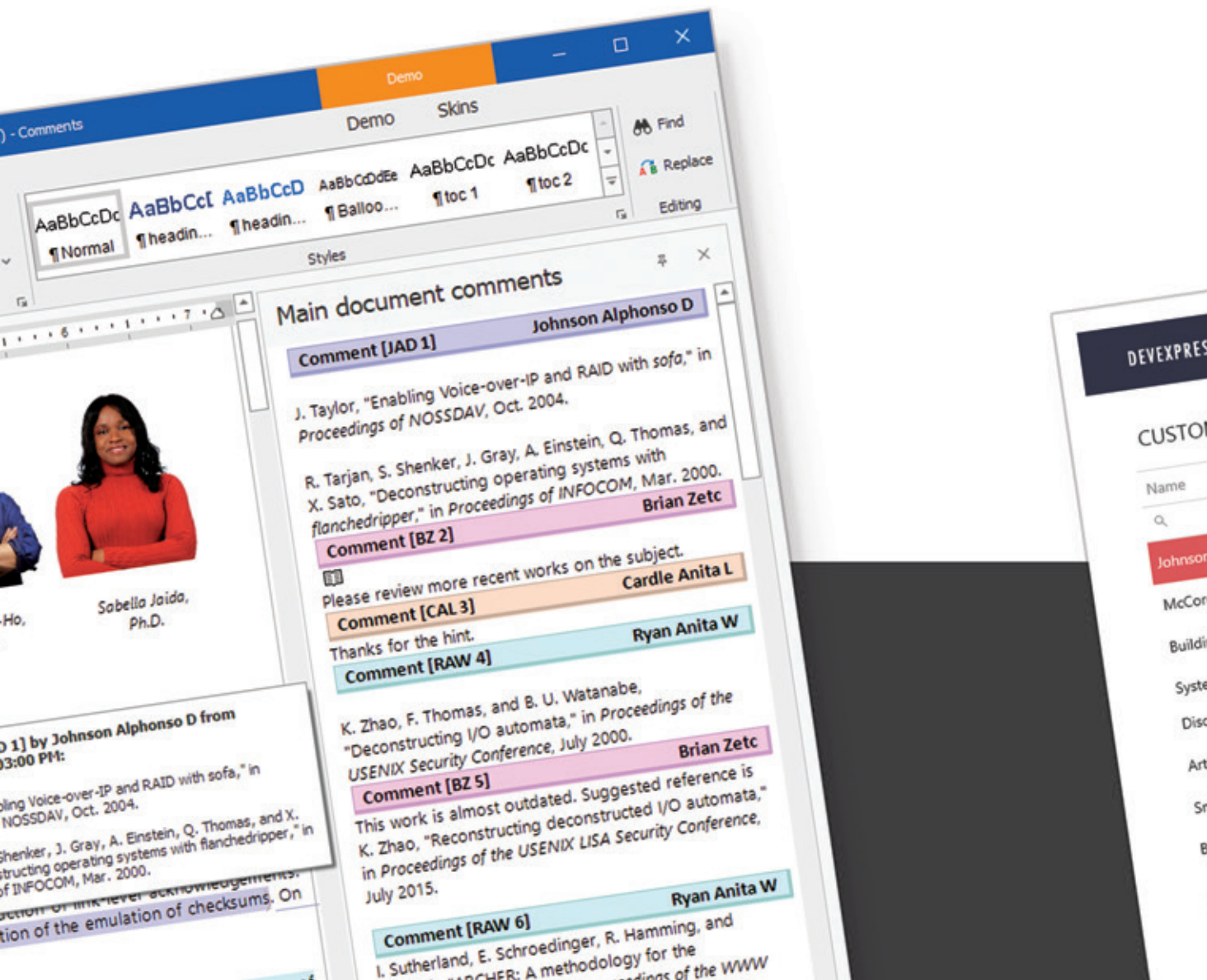
When rich text editing is a must and you need to replicate key features found inside Microsoft Word® (from mail merge and tables, to floating objects and document comments), look no further than the DevExpress Rich Text Editor Control. The library's comprehensive feature set includes printing and PDF export support and the availability of DevExpress Visual Studio project templates means you can incorporate our rich text editors in your next .NET app within minutes of install.

.NET Framework	WinForms / WPF / WebForms / MVC
.NET Core 3.0	WinForms / WPF / ASP.NET
JavaScript	jQuery / Angular / React / Vue

Your Next Great Text Editor Starts @DevExpress

See how you can introduce Word® compatible rich text editing in your app.

devexpress.com/word



Charting & Analytics

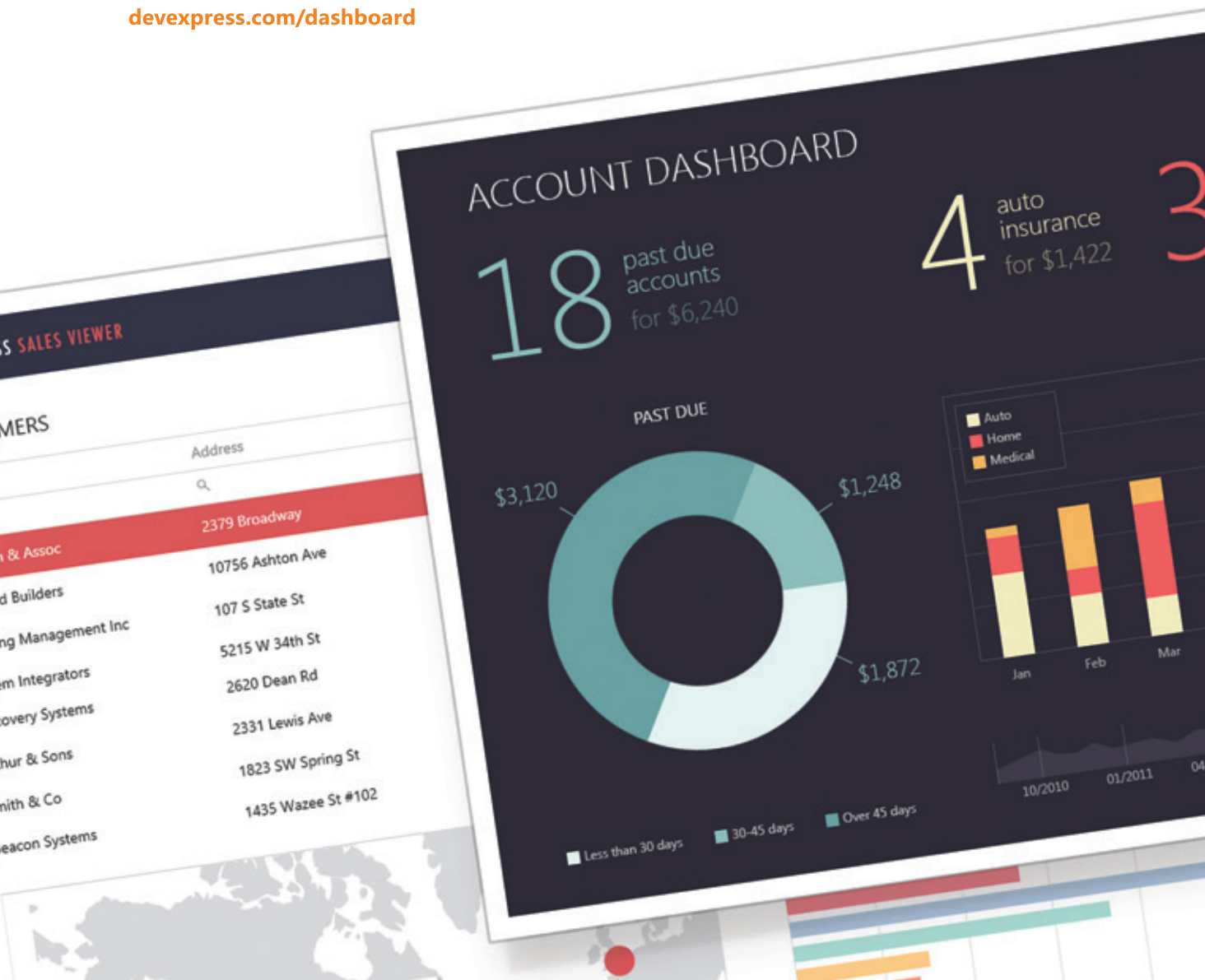
From zero to dashboard in record time.

With our integrated suite of charts, pivot grids, and multi-purpose dashboard widgets, you'll create information-rich decision support systems that are optimized for Windows, the web and your favorite mobile device. The UI components inside our subscriptions help you deliver adaptable, interactive and touch-enabled user experiences that can address a broad range of use-case scenarios. And perhaps best of all, the dashboards you build with DevExpress Universal can be distributed royalty-free.

.NET Framework	WinForms / WPF / WebForms / MVC / UWP
.NET Core 3.0	WinForms / WPF / ASP.NET / Blazor
JavaScript	jQuery / Angular / React / Vue
Mobile	iOS / Android / Xamarin

Your Next Great Dashboard Starts @DevExpress

See how you can build future proof, enterprise-grade decision support systems.
devexpress.com/dashboard



READY • SET • GO

Download Your Free 30-Day Trial Today

devexpress.com/try

DevExpress®

The triple-quoted string is what's often called a "here-doc" multi-line string literal—it will capture everything inside the pair of triple quotes, including whitespace, making it useful for documentation purposes. Additionally, Python will do something quite special to this string—it will capture it into a property on the function itself, and make it discoverable at runtime via the "__doc__" property name. To see what this means, try calling this bit of Python after the function has been defined:

```
print(theKnightsWhoSayNi.__doc__)
```

Lo and behold—helpful documentation about the function is printed to the command line! And it turns out that this "__doc__" is not the only interesting thing on the function—if you do a dir on the function name, a whole slew of interesting things (properly called "attributes") appear:

```
['__annotations__', '__call__', '__class__', '__closure__', '__code__',
'__defaults__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__',
'__format__', '__ge__', '__get__', '__getattribute__', '__globals__',
'__gt__', '__hash__', '__init__', '__init_subclass__', '__kwdefaults__',
'__le__', '__lt__', '__module__', '__name__', '__ne__', '__new__',
'__qualname__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__',
'__sizeof__', '__str__', '__subclasshook__']
```

Clearly, Python is doing a fair bit of interesting organization and structuring of functions under the hood.

Customizing Arguments

In some cases, you want arguments to be optional, meaning you want to provide a default value to the parameter in the event the caller doesn't provide one. This is done by providing said value in the function definition:

```
def theKnightsWhoSayNi(gift = None):
  """In order to continue your quest for Camelot, you must
  bring the Knights gifts. Do not mess with these guys, for
  they say 'Ni' at the slightest provocation!"""
  if gift == "shrubbery":
    return ("You must now cut down the tallest tree"
      "in the forest... with... A HERRING!!!!")
  else:
    return "You must bring us... A SHRUBBERY!!!"
```

Now the function can be invoked with either one parameter or none. None, by the way, is the closest equivalent Python has to null or nil in other languages.

Python also has what they call "keyword parameters," which allows you to use named parameters in function calls instead of (or in addition to) positional ones. This means that if you have a function defined with a few default values, like so:

```
def parrot(voltage, state='a stiff', action='voom', type='Norwegian Blue'):
  print("-- This parrot wouldn't", action, end=' ')
  print("if you put", voltage, "volts through it.")
  print("-- Lovely plumage, the", type)
  print("-- It's", state, "!")
```

You can then call the function in a variety of different ways, either by using positional arguments (the normal way) or by keyword arguments, like this:

```
parrot(1000)
parrot(type='Turkish Red', action='caw', voltage=1000)
```

In any case, all non-default parameters must have a value, and keyword arguments can only appear after all positional arguments do. If you prefer to pass a dictionary for the arguments, put two asterisks in front of the dictionary parameter at the point of call:

```
d = {"voltage": "a million", "state":"bleedin' demised"}
parrot(**d)
```

Python also allows for arbitrary argument lists, akin to the "params" modifier from C#, if the parameter is declared with an asterisk preceding its name. This will then capture all positional arguments after that point into a single list for processing, like so:

```
def order(bread, sides, *fixings):
  print("Making you a Spam-on-", bread)
  print("  with", sides)
  for f in fixings:
    print("  and", f)
  print("Enjoy!")

order("white", "pickles")
order("wheat", "radishes", "pickles", "chips", "pint o' lager")
```

Nothing prevents you from writing all your functions to take one arbitrary argument list, just as you can do in ECMAScript, but most of the time that's not idiomatic Python and will likely get somebody to chase you with a herring.

Anonymous Functions

Python also supports functions-as-first-class-citizens out of the box, in terms of passing a function as a parameter, such as what you might use in the map global function, which executes a function over each element of a list:

```
items = ["bread", "eggs", "milk"]
def spametize(item): return "spam"

spamCafeOrder = map(spametize, items)
print("We need to go to the market and get:")
for it in spamCafeOrder: print(it)
```

In some cases, you want arguments to be optional, meaning you want to provide a default value to the parameter in the event the caller doesn't provide one.

But when working with functions, it's often useful to define a function anonymously. In Python, you do this via the "lambda" keyword, which replaces the "def" and function name in the definition:

```
def add(lhs, rhs): return lhs + rhs
def makeAdder(lhs):
  return lambda rhs: add(lhs, rhs)

addFive = makeAdder(5)
print(addFive(12)) # prints 17
```

Unlike other languages, Python insists that lambda functions be restricted to small-use scenarios: A given lambda-defined function can only consist of a single expression (such as the add call in the preceding example).

Wrapping Up

Python functions are among the core building blocks of the platform, and every bit as important as classes and objects. Unlike some of the "traditional" object-oriented languages (C#/C++/Java), Python not only doesn't fear global functions, it embraces them as a key design tool.

Because Python has lambdas and first-class functions, many people will categorize Python as a functional language (on the order of languages like Haskell or F#), but I do not. Python lacks a number of the features that a traditional functional language will have (like partial application of functions or immutability by default). Although you can pass functions around as objects, you can do the same thing in a number of other languages, so if Python is functional, so is C. And, trust me, nobody wants to go there.

Farewell, All

As you have either heard or read, this installment marks the end of this column, and this magazine. To say that this makes me emotional is quite the understatement; 20 years ago, I was the feverish consumer of every issue of *Microsoft Systems Journal*, drinking up every bit of wisdom and lore it offered like the proverbial thirsty man in the desert. Then I got to write for the magazine, and I cannot tell you how amazing that felt; it was a bucket-list item checked off, to be sure.

> Python functions are among the core building blocks of the platform, and every bit as important as classes and objects.

For close to a decade, you've been giving me the privilege of exploring topics with you, and I hope you've enjoyed that ride as much as I have. From Mongo and Cassandra to Naked Objects, with stops at MEAN and Multiparadigmatic Design along the way, I've had a ton of fun learning and teaching and exploring and, even, when the mood called for it, having a little fun (the LOLCODE article will be, by far, my favorite column installment I will ever write).

For me and the other *MSDN Magazine* authors, our journey with you through the magazine ends here. But the journey as a whole continues: All of us authors, we're out there, in the world, and we're just as approachable for questions and discussion as we were here. If you find one of us at a conference, don't be a stranger. Come on up, say hi, introduce yourself and tell us what you're working on. The two-way street that this magazine created between us isn't shut down when the magazine shutters its doors—it just needs to take a different form.

All good things in time must come to an end, and so have I. Please, as you go about your career and life, always remember:

Happy coding! ■

Ted Neward *is a Seattle-based polytechnology consultant, speaker and mentor. He has written a ton of articles, authored and co-authored a dozen books, and speaks all over the world. Reach him at ted@tedneward.com or read his blog at blogs.tedneward.com.*

Thanks to the following technical expert for reviewing this article: *Harry Pierson*

Visual Studio LIVE!
EXPERT SOLUTIONS FOR ENTERPRISE DEVELOPERS

NASHVILLE CODE TRIP 1993 2020
A DEVELOPER'S ROAD MAP

#VSLive

NASHVILLE
GRAND OL' CODE

May 17-21, 2020
Loews Vanderbilt Hotel

Track Topics to Include:

- Artificial Intelligence, Data and Machine Learning
- Cloud Containers and Microservices
- Delivery and Deployment
- Developing New Experiences
- DevOps in the Spotlight
- Full Stack Web Development
- .NET Core and More

Don't miss out y'all! Register by March 27 and save $500.

SUPPORTED BY

Microsoft | Visual Studio | Visual Studio MAGAZINE

PRODUCED BY

CONVERGE 360 An 1105 MEDIA company

VSLive.com/nashville

FRANK LA VIGNE

Exploring Facial Detection and Recognition

Humans are uniquely adept at detecting and recognizing faces, and face recognition is among the first cognitive skills that humans develop. In fact, we're so good at this that we often perceive faces where there are none—a phenomenon known as pareidolia. In engineering terms, face detection algorithms in the human brain are prone to false positives. It's taken computer science a considerable amount of time to develop algorithms that can detect faces and accurately identify people based solely on their facial appearance, but now that it's happened, it's provoking an important ethical and legal debate.

This isn't my first dance with face recognition. In my November 2016 Modern Apps column (msdn.com/magazine/mt788628), I used a function built into the Universal Windows Platform (UWP) to detect faces in images captured by a camera. Exploring how the built-in detection system actually recognized faces was beyond the scope of that article at the time, but now I'll explore in some detail how facial detection and recognition works. Yes, the Cognitive Services Face API (bit.ly/2l0d5y1) can perform all these tasks for you, but I wanted to look closer at how the underlying mechanisms work. Doing so gave me a greater appreciation of the Face API.

Detecting Faces

It's important to point out two distinct terms that are often used interchangeably: face detection and face recognition. Face detection, as the name implies, is limited to detecting the presence of faces in an image. Face recognition involves discerning unique facial characteristics (such as location and shape of the eyes, nose, mouth) to identify individuals based solely on their facial appearance.

Finding faces in images was a problem that eluded computer science until the early 2000s, when researchers Paul Viola and Michael Jones pioneered an algorithm that now bears their name. The Viola-Jones algorithm scans an image using a rectangular filter looking for contrasting patterns of light and dark. While prone to false positives, Viola-Jones is fast and ideal for low-power and battery-driven devices. For an in-depth explanation of the Viola-Jones algorithm, I recommend the following YouTube video: youtu.be/uEJ71VlUmMQ.

Another approach to detecting faces is the Histogram of Oriented Gradients (HOG), which analyzes each pixel in an image and its immediate neighboring pixels looking for changes from light to dark. This captures two points of data: the magnitude and the direction of the change from light to dark. This has the end effect of finding the edges in an image and ignoring smoother areas where there are little differences between pixels. The next step is to break down the image into smaller units, say 16 x 16 pixels, and then average out the magnitude and direction values. This reduces the data significantly and provides a bit of noise reduction. Noise, in this context, means any changes from light to dark that aren't of interest to the algorithm. By taking an average over a set of pixels, we blur out the minor variations in an image, such as subtle shadows, and are left with only the major differences that tend to denote the edges of objects.

For a more in-depth look at this algorithm, check out this tutorial: bit.ly/2mkjCne. While it focuses on using HOG in the context of the OpenCV library, the algorithm and the mathematics behind it are the same.

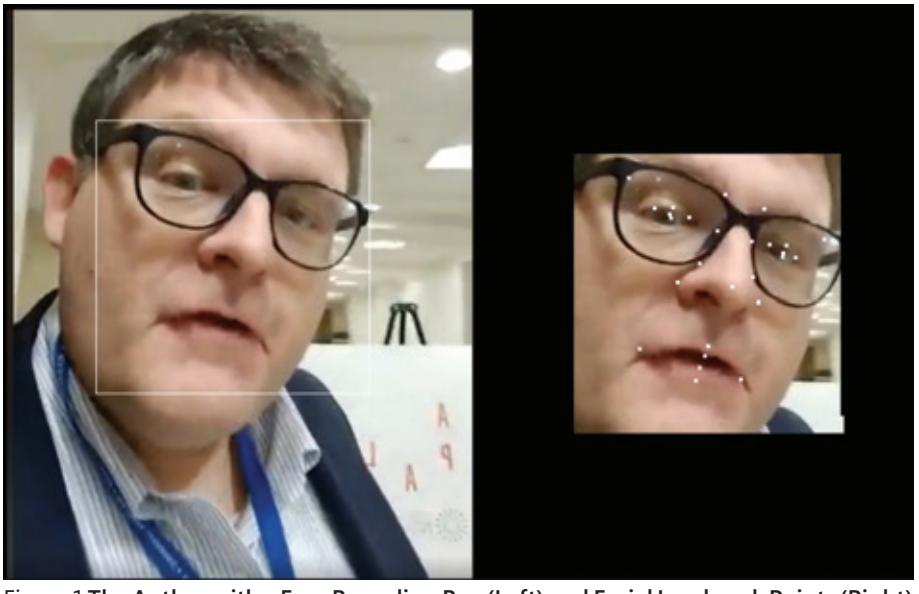


Figure 1 **The Author with a Face Bounding Box (Left) and Facial Landmark Points (Right)**

Code download available at bit.ly/2kKIxjC.

Thank you, MSDN Magazine.

We'll miss you.

Finding Face Landmarks

Once a face is detected, the next step is to determine the coordinates of common facial features in the image. There are 68 landmark points on the human face that are of interest to most face detection algorithms. These points include the nose, the mouth, eyebrows, jaw and more. **Figure 1** depicts highlighted facial landmark points.

This level of detail provides data for two useful purposes. First, the algorithm can determine the orientation of the face. In a two-dimensional image, the locations of the 68 landmark points can get distorted by the roll, tilt and angle of the face. The algorithm's ability to infer this information is how popular messaging apps can do things like place virtual sunglasses over people's eyes or apply makeup on a face and sync its movements with the person as he or she moves. Second, this collection of facial landmark points is used to identify an individual. The distance between these points is a unique characteristic and algorithms can be trained to recognize an individual to a degree of accuracy rivaling humans.

Coding a Face Detection System

Let's start by creating a Python 3 notebook on your preferred platform (I covered Jupyter Notebooks in a previous column at msdn.com/magazine/mt829269). Create an empty cell, enter the following code to enable inline display of images and execute the cell:

```
%matplotlib inline
```

There are multiple libraries that perform face detection and face recognition. For this article, I chose to work with the face_recognition library at pypi.org/project/face_recognition. Create a new cell and enter the following code to install it:

```
! pip install face_recognition
```

This process may take a few moments as the package downloads, compiles and installs. Enter the following code into a new cell and execute it to import the required libraries:

```
from matplotlib.pyplot import imshow
import numpy as np
import PIL.Image
import PIL.ImageDraw
import face_recognition
```

Next, enter the code in **Figure 2** to create a function that takes an image file and runs it through the face_recognition library.

After executing that code, create a new cell and enter the following:

```
findFaces("frank.jpg")
```

The output should read:

```
1 face(s) in this image
Face coordinates: Top: 172, Left: 171, Bottom: 726, Right: 726
```

Feel free to test the algorithm with your own images. In the project on the Azure Notebook service, I have several images in the project directory to test with, including a crowd image with multiple faces.

Next, enter the following code to display all the face landmarks in an image:

```
faceImage = face_recognition.load_image_file("frank.jpg")
face_landmarks = face_recognition.face_landmarks(image)
print(face_landmarks)
```

The result should display sets of coordinates labeled "left eyebrow," "nose tip" and so on. Again, feel free to experiment with your own images.

As stated earlier, detecting faces and finding landmarks are the precursors to face recognition. Let's now use the face_recognition library to compare two images—frank.jpg and frank2.jpg—to see if they're the same person. Enter the following code into a new cell and execute it:

```
known_image = face_recognition.load_image_file("frank.jpg")
mystery_image = face_recognition.load_image_file("frank2.jpg")

frank_encoding = face_recognition.face_encodings(known_image)[0]
mystery_encoding = face_recognition.face_encodings(mystery_image)[0]

results = face_recognition.compare_faces([frank_encoding], mystery_encoding)

print (results)
```

Not surprisingly (as both images are of me) the code returns a result with the Boolean value of true. Next, enter the following code into a new cell and execute it:

```
mystery_image2 = face_recognition.load_image_file("andy.jpg")
mystery_encoding2 = face_recognition.face_encodings(mystery_image2)[0]

results = face_recognition.compare_faces([frank_encoding], mystery_encoding2)

print (results)
```

In this instance, I've introduced the image (andy.jpg) of another person, and compared it to the image of frank.jpg. And no surprise, the result comes back as false. Thus far we've only compared images containing one face. What about more complicated images? Let's now apply facial recognition to images depicting a crowd of people. Enter the following code into a new cell and execute it:

```
crowd_image = face_recognition.load_image_file("crowd.jpg")
crowd_encoding = face_recognition.face_encodings(crowd_image)
for encoding in crowd_encoding:
  is_frank_in_the_crowd = face_recognition.compare_faces([frank_encoding], encoding)
  print(is_frank_in_the_crowd)
```

The answer comes back as false 21 times, which is correct as there are 21 faces detected in that image and I am not in the picture. Feel free to experiment with various pictures of your own to see what kind of results you can get.

Wrapping Up

Our faces, by definition, are personally identifiable information (PII). So it comes as little surprise that work in the field of face recognition has stirred up controversy. The Viola-Jones algorithm was a landmark discovery that led to widespread innovation in everything from digital cameras to surveillance systems, yielding deep concerns about privacy and civil liberties (to the point that some jurisdictions have banned use of face recognition by law enforcement). Given our relentless march toward a more connected and data-driven society, concerns around this technology—especially in the context of AI-driven systems—will only intensify.

Figure 2 **Code for the findFaces Function**

```
def findFaces(imageName):
  image = face_recognition.load_image_file(imageName)

  face_locations = face_recognition.face_locations(image)

  number_of_faces = len(face_locations)
  print("{} face(s) in this image".format(number_of_faces))

  pil_image = PIL.Image.fromarray(image)

  for face_location in face_locations:

    top, right, bottom, left = face_location
    print("Face coordinates: Top: {}, Left: {}, Bottom: {},
      Right: {}".format(top, left, bottom, right))

  imshow(np.asarray(pil_image))
```

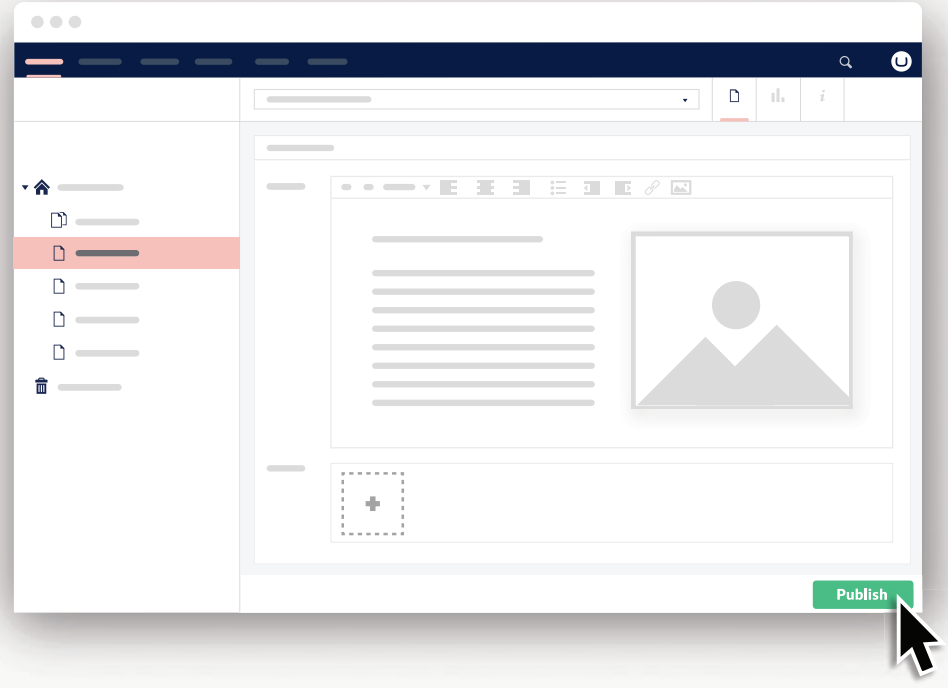
Say hello to full flexibility

Say hello to **Umbraco** - the Content Management System voted #1 by your peers

Our Open Source .NET CMS is flexible and developer-friendly, which gives you the power to develop websites that will impress your clients, boss and coworkers alike.

Umbraco Cloud is the perfect hosting solution for your Umbraco site - and so much more. Added features help you save hours on tedious tasks so you can spend your time on building amazing websites.

Over 225,000 active developers are part of our worldwide community who help us shape and develop the CMS. So if you need a helping hand, you're never on your own.



umbraco.com/try/

umbraco

The Friendly CMS

Visual Studio MAGAZINE READER'S CHOICE 2020

Saying Goodbye

These last few years have been an interesting personal and professional journey for me, as I transitioned from a smart client developer to a machine learning engineer. I discovered that much of the training material in the artificial intelligence (AI) space was geared toward academic researchers and those with extensive experience in advanced mathematics. There just wasn't much for software engineers to grab hold of. This inspired me to retire the Modern Apps column and its coverage of UWP development, and re-launch as Artificially Intelligent. The goal: To explore the realms of data science and AI from a software engineer's point of view.

This column will be coming to an end, but I plan to continue my work making AI and data science more approachable to software developers. I'll continue to post articles on my blog and to podcast at DataDriven. Recently, I started hosting virtual summits, one-day virtual events focused on a particular topic. And I have other projects in the works to continue my mission of developer training and empowerment. For a full list of activities and resources, visit **franksworld.com/msdn** for offers, links and discounts just for readers of this column.

It has been a distinct honor and privilege to write for *MSDN Magazine* these last several years. I've been amazed and delighted to be recognized for my column by people at an event or customer meeting, and to hear how much they learned from the column over the years. Finally, I would like to extend my deep gratitude to Rachel Appel, whose column I took over in 2016, and to Michael Desmond for being such a great (and patient) editor.

The ethical and political issues around facial recognition may be complicated, but the algorithms enabling basic face detection and recognition are not. In fact, much of what this article explored could've been easily accomplished with the Cognitive Services Face API via a few simple REST calls. However, I feel it's important for software engineers to have a deeper knowledge of the algorithmic underpinnings of these tools and to appreciate the work that went into creating the Face API. Additionally, knowing more about the underlying mathematical principles can help developers identify edge cases they may encounter.

In this article I touched upon the rich and fascinating subject of facial recognition. This is very much at the forefront of AI research. Much is still being done in this field to reduce the rate of false positives and mitigate biases. And researchers continue to make discoveries, for instance the recent finding that certain patterns and colors in face paint can hinder many face detection systems (bit.ly/2mknOhW). ■

Frank La Vigne *works at Microsoft as an AI Technology Solutions Professional where he helps companies achieve more by getting the most out of their data with analytics and AI. He also 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 Microsoft technical expert for reviewing this article: *Andy Leonard*

Visual Studio LIVE!
EXPERT SOLUTIONS FOR ENTERPRISE DEVELOPERS

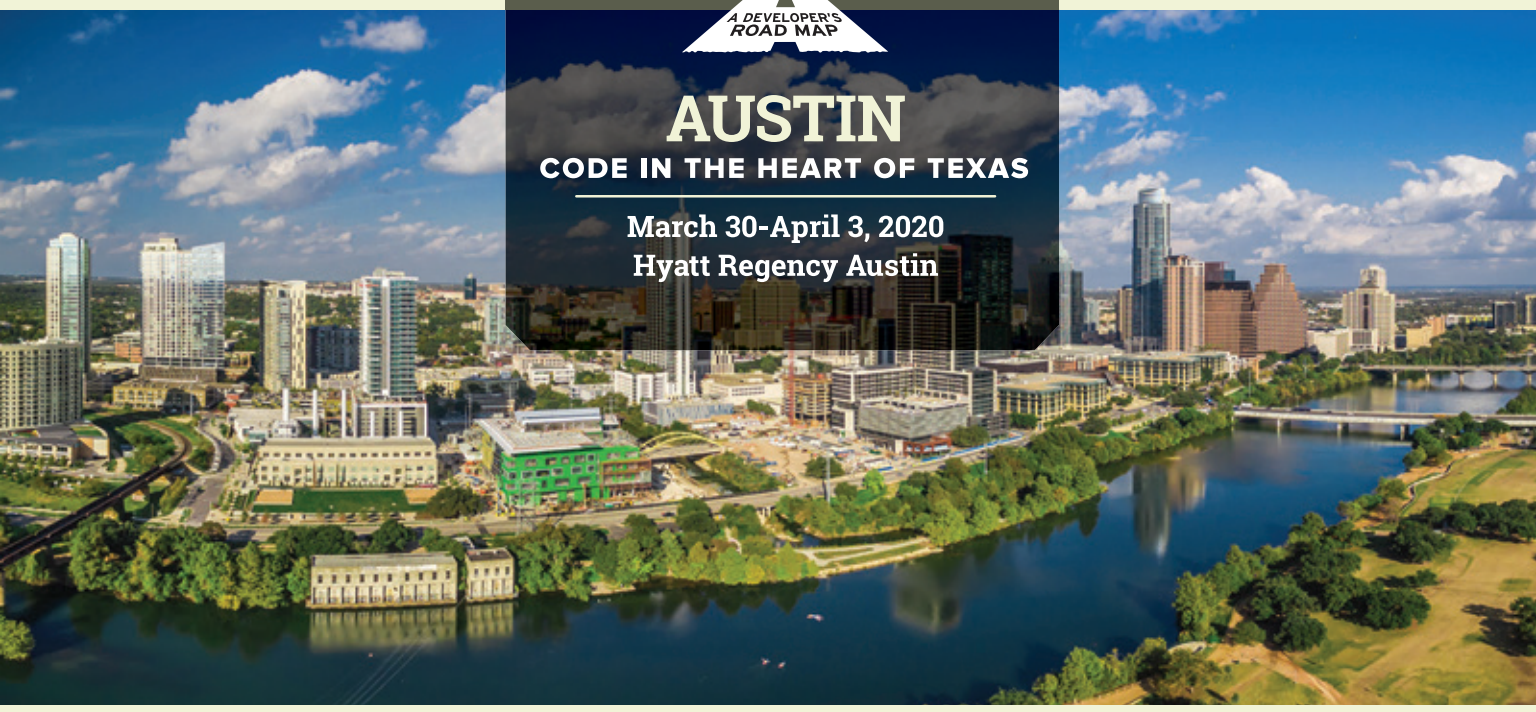
AUSTIN 1993 CODE TRIP 2020
A DEVELOPER'S ROAD MAP

#VSLive

AUSTIN

CODE IN THE HEART OF TEXAS

March 30-April 3, 2020
Hyatt Regency Austin



Track Topics to Include:

- Artificial Intelligence, Data and Machine Learning
- Cloud Containers and Microservices
- Delivery and Deployment
- Developing New Experiences
- DevOps in the Spotlight
- Full Stack Web Development
- .NET Core and More

Don't miss out on Texas-sized savings! Register by January 31 and save $500.

SUPPORTED BY

Microsoft | Visual Studio | Visual Studio MAGAZINE

PRODUCED BY

CONVERGE 360 An 1105 MEDIA company

VSLive.com/austin

Preparing for the Exponential Technology Revolution

Mark Michaelis

The world you know is about to change in profound and radical ways. A historic confluence of emerging technologies, powered by ubiquitous connectivity and advances like artificial intelligence (AI), are poised to complement and catalyze each other to change the way we work, play, relate and live.

Right about now, you probably know this isn't going to be your usual *MSDN Magazine* article. While the magazine remains committed to code-level guidance for developers, this final issue offers an opportunity to ponder the future and how today's technology will shape and inform it. We stand at an inflection point in technology evolution, and we need to understand that it involves exponential change that will yield a world very different from the one we know. But it's a chance, also, to explore the role and responsibility that we as developers will have as we engage with and enable the coming tide of innovation. There are moral and ethical choices ahead of us, also, and grappling with them will be a core challenge of the coming age.

This article discusses:

- The confluence of numerous emerging exponential technologies
- The challenge of identifying exponential technologies
- The unique role developers play in enabling exponential technologies and determining their societal impacts

Technologies discussed:

Exponential Technologies, Responsible Innovation

Needless to say, readers of *MSDN Magazine* are uniquely impacted by the accelerating technological forces changing our world. Developers like you right now are writing the code that drive the systems that will shape our future. In this, the final issue of *MSDN Magazine*, I can think of no more fitting topic for exploration.

Understanding Exponential Technologies

To grasp the coming technology revolution, it's necessary to understand the concept of an exponential technology. Consider for example the graph in **Figure 1**, which tracks the growth in the number of photos taken annually worldwide. Since 2000, the emergence of digital cameras, and then camera-equipped cell phones, has driven skyrocketing growth. In 2000, about 86 billion photos were taken. Twelve years later, that number was 380 billion.

The problem with charts like this, and the numbers underlying them, is that they illuminate exponential trends in hindsight, only after they've already happened. Forecasting exponential growth is difficult because we're tuned for linear thinking—we struggle to grasp the magnitude of exponential trends.

To recognize exponential trends early—when growth still looks linear—you need to identify the key technology improvement that drives it. In the case of digital photography—and many other exponential technologies—that driver is digitization.

Digitization is closely associated with exponential technology because it tends toward zero cost. The more digital photos you take, the closer the cost per photo approaches free. Even digital storage isn't far from a zero-cost item—there are numerous cloud providers

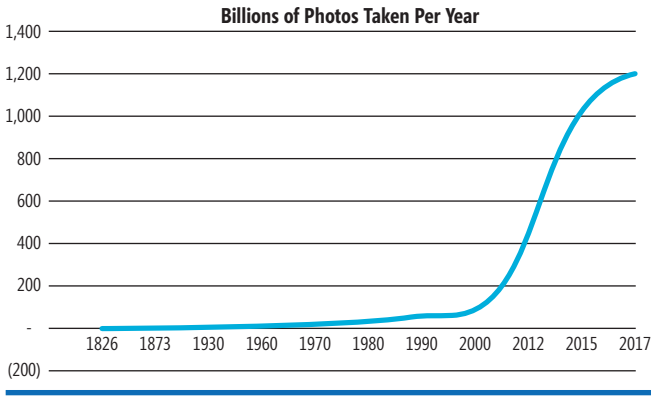


Figure 1 **The Exponential Growth of Photography**

offering storage that's close to free (especially if the provider is licensed to advertise or mine the information posted).

Another key characteristic of exponential technologies is democratization—making the technology ubiquitous. In the case of photography, this occurred because of the parallel adoption of mobile phones—another exponential technology. (For more on this dynamic, consider exploring the work of Salim Ismail and Steven Kotler, who both identify characteristics to look for in identifying exponential technologies.)

Figure 2 offers a small sample of some of today's exponential technologies. As with photography and mobile phones, what's amazing is that each is likely exponential on its own, but together they amplify each other, making the doubling time even shorter.

The breadth of even the short list in **Figure 2** is almost shocking. And when you consider the impacts of all these technology areas converging at once, it raises the question: Are we experiencing another industrial revolution?

Industrial Revolution or Inflection Point

Historically, industrial revolutions are characterized by changes in labor—the way we work. Whether it was the First Industrial Revolution (steam and mechanization), the Second Industrial Revolution (electric power and mass production) or the Third Industrial Revolution (digitization and information technology), they all involved a radical change in labor.

Figure 2 **Exponential Technology Areas**

Material Science	Energy Storage	Autonomous Transportation (cars/ trucks, drones, etc.)
Internet of Things (IoT)	Quantum Computing	Nanotechnology
Artificial Intelligence/ Machine Learning	Blockchain	Bio-Technology/ Medicine
Robotics	Computers	Virtual Reality
3D Printing	Cloud Computing	Neuroscience

Figure 3 **Net Income Per Employee for Top Tech Firms**

	No. of Employees	Net Income Per Employee
Facebook	25,105	$634,694.00
Apple	92,600	$393,097.00
Microsoft	118,000	$171,000.00
Alphabet (Google)	53,861	$158,057.00

Today, the news about labor is peppered with concerns about AI and autonomous robots (in which I include autonomous transportation) and how these technologies will replace today's work force. Research by the McKinsey Global Institute predicts that 45 percent of current work activities could be automated by existing technology right now. It also finds that about one-third of the tasks in 60 percent of existing jobs can be performed by computers. These are premonitions of dire unemployment.

Of course, similar predictions occurred with the previous industrial revolutions, and all proved inaccurate with time. While work was radically affected in all cases, especially when the technology was democratized, the technology that triggered each industrial revolution generally enabled people to move on to better jobs overall. In short, labor improved. Is today's technology any different?

> The problem with charts like this, and the numbers underlying them, is that they illuminate exponential trends in hindsight, only after they've already happened.

Staying with the labor theme, consider the profits of companies delivering exponential technologies. For a company like Amazon the number of employees doesn't correlate to an increase in sales, which may increase exponentially while employment only increases linearly. Likewise, companies in exponential markets can see net income per employee (NIPE) increase exponentially. Consider, for example, the NIPE of the top four tech companies in 2018, as reported by CSIMarket, shown in **Figure 3**. By contrast, the average NIPE in 1990 for the top three automobile manufactures was just less than $60,000, when adjusted for inflation.

At the same time, companies are favoring independent contractors over full-time employees. (This dynamic may change as employment law steps in to protect workers, such as the recently passed Assembly Bill 5 in California, which requires that app-based companies treat their contractors like employees.) Regardless, the breadth of disruption across major industries such as print media, music, television, transportation, hotels, banking and agriculture is breathtaking.

These are all significant changes in labor, but are they enough to credit this decade with the introduction of a fourth industrial revolution, as suggested by Klaus Schwab, head of the World Economic Forum and author of the book, "The Fourth Industrial Revolution" (Penguin Group, 2017)? Or are these trends simply a continuation of the the digital revolution? That is, perhaps, a question best left to historians to answer, but today's technology revolution does present some important differences from past cycles. Consider:

- The quantity, concurrency and fusion of exponential technologies as they potentially amplify each other. Example: Energy storage amplifies drones technology.
- The exponentially increasing adoption of cyber physical devices connected to the Internet. Example: home automation devices.
- The core value of data mined from technology, be it via devices, online services or other technologies. Example: The McKinsey report estimates revenue from data mined and monetized from connected cars could reach $750 billion in 2030.
- The exponential velocity of change. Example: The incredibly rapid acceptance and ubiquitous adoption of smartphones and soon after cloud services.

"Get thee to the cloud" is obviously advice you've heard before, but it becomes imperative if you hope to capitalize on the exponential growth happening in cloud technologies.

In fact, we're seeing transformative change for entire systems and industries, such as in transportation, travel and purchasing. What's more, these paradigm shifts are changing us, both as individuals and as communities and nations. Smartphones, for instance, have already revolutionized personal interaction and relationships.

Confronting Questions

Whether we identify the current decade as the Fourth Industrial Revolution, or simply an inflection point in the digital revolution, there are significant questions to confront. And as programmers, we're in the unique position of implementing the software that will ultimately control the exponential technologies.

First, there are the obvious ethical questions about many of the innovations in the offing—genetic engineering is an easy example. While I'm surprised and concerned by how few organizations employ ethicists on their teams, the fact is any employee can fill that role simply by raising ethical questions. And there have been some employee protests in recent years that reflect this commitment.

Second, we can't innovate without considering the issue of governance. How do we regulate exponential technologies that may be used for good or ill? You can use facial recognition to tag your personal photo collection just as easily as governments can use it to track and control populations. Similarly, drones can be used to carry medicines to the sick following a hurricane, but they can be equally effective as weapons.

Almost any technology can be used for good or evil, but the challenge of exponential technology is the potential for exponential consequences. Whether companies self-regulate or the law provides boundaries, either approach will struggle to keep pace with rapid change. Autonomous driving, social media propaganda and ethnic/gender bias in machine learning have already emerged as immediate challenges.

Finally, we should consider what motivates our innovation. Capitalism was a core driver of the industrial revolutions of the past and it yielded profound progress, but it also produced troubling inequities. Now, as we step forward into a world reshaped by exponential technologies, we have the opportunity to improve humanity and to ensure the economic and social benefits of our advances are felt by all.

Next Steps

When you consider the pace and breadth of technology advancement, it's clear we live in a time like no other. As software developers, we're in a unique position to influence the future. Whether you're engaged with exponential technologies in the physical sciences (medicine, energy storage, bio engineering) or in the computer sciences (AI, cloud, Big Data, blockchain), you have opportunities to have an impact. And for those not yet leveraging these technologies, you should be urging your organizations to catch up. Organizations that fail to engage exponential technologies early in the curve alongside their competitors court the risk of being left exponentially behind.

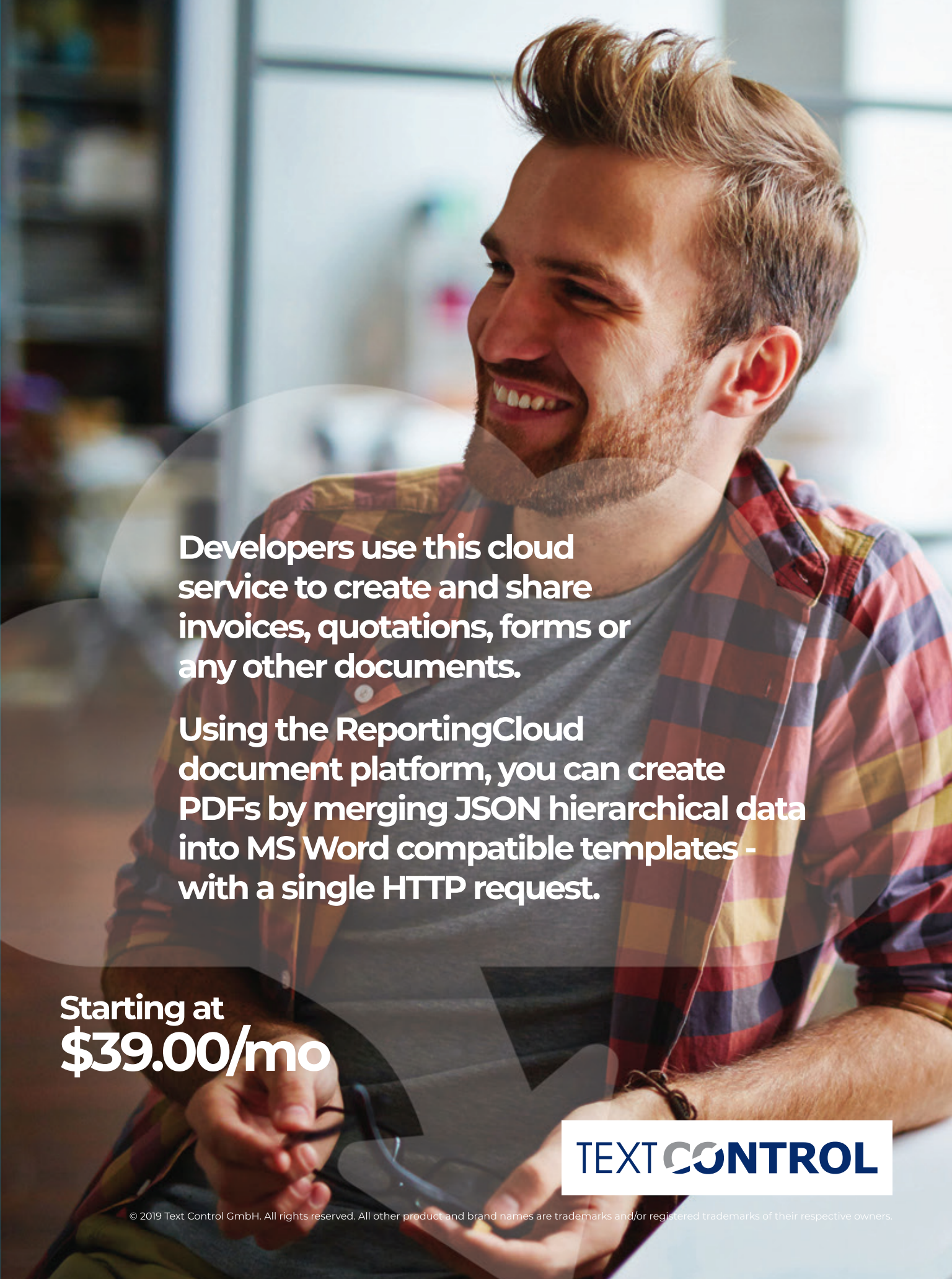
Now is the time to explore opportunities to digitize your product. Research how to information-enable your offerings so the data becomes a profit center. At the same time, evaluate the cloud and at a minimum move there for Platform-as-a-Service (PaaS) implementations. "Get thee to the cloud" is obviously advice you've heard before, but it becomes imperative if you hope to capitalize on the exponential growth happening in cloud technologies. And nowhere is this growth more dramatic than in the field of AI and machine learning.

These are premonitions of dire unemployment.

Finally, don't be afraid to think big. Look beyond leveraging existing technologies and budget time to ideate about your next pivot. Instead of Uber replacing taxi drivers, think of autonomous driving replacing drivers entirely. Consider forming an ideation team within your company that allocates time each week to complain about frustrations and concoct solutions.

In summary, capture the potential of today's exponential technology, focus on how you can leverage or even surpass it to make the world a better place, and then—just do it. ■

Mark Michaelis *is founder of IntelliTect, where he serves as its chief technical architect and trainer. For nearly two decades he has been a Microsoft MVP, and a Microsoft Regional Director since 2007. Michaelis serves on several Microsoft software design review teams, including C#, Microsoft Azure, SharePoint and Visual Studio ALM. He speaks at developer conferences and has written numerous books including his most recent, "Essential C# 7.0 (6th Edition)" (itl.tc/EssentialCSharp). Contact him on Facebook at facebook.com/Mark.Michaelis, on his blog at IntelliTect.com/Mark, on Twitter: @markmichaelis or via e-mail at mark@IntelliTect.com.*



Iterating with Async Enumerables in C# 8

Stephen Toub

Since the beginning of .NET, enumeration of collections has been the bread-and-butter of many programs. The non-generic System.Collections.IEnumerable interface enabled code to retrieve a System.Collections.IEnumerator, which in turn provided the basic functionality of MoveNext and Current for forward iterating through each element in the source collection. The C# language simplified this iteration further via the foreach keyword:

```
IEnumerable src = ...;
foreach (int item in src) Use(item);
```

When .NET Framework 2.0 came around with generics, System.Collections.Generic.IEnumerable<T> was introduced, enabling retrieving a System.Collections.Generic.IEnumerator<T> to support strongly typed iteration, a boon for both productivity and performance (due in large part to avoiding boxing):

```
IEnumerable<int> src = ...;
foreach (int item in src) Use(item);
```

As with the non-generic interface, this use of foreach is transformed by the compiler into calls on the underlying interfaces:

```
IEnumerable<int> src = ...;
IEnumerator<int> e = src.GetEnumerator();
try
{
  while (e.MoveNext()) Use(e.Current);
}
finally { if (e != null) e.Dispose(); }
```

In addition, C# 2.0 introduced iterators, which make it simple for developers to use normal control flow constructs to author custom enumerables, with the compiler rewriting developer methods that employ "yield return" statements into state machines suitable to implement IEnumerable<T> and IEnumerator<T>:

```
static IEnumerable<int> Range(int start, int count)
{
  for (int i = 0; i < count; i++)
  yield return start + i;
}
...
foreach (int item in Range(10, 3))
  Console.Write(item + " "); // prints 10 11 12
```

(This example shows using "yield return" to create an IEnumerable<T>, but the language also supports using IEnumerator<T> instead, in which case it's equivalent to an IEnumerable<T> being produced and then GetEnumerator being called on the result.)

Fast forward to .NET Framework 4, which introduced the System.Threading.Tasks.Task and Task<T> types, and .NET Framework 4.5 and C# 5, which introduced the async and await keywords to drastically simplify asynchronous programming. Instead of having to write complicated callback-based "spaghetti" code, developers could again use normal control flow constructs to author their asynchronous operations, with the compiler rewriting the developers' async methods that employ await expressions into state machines that generate Tasks and internally use callbacks:

```
static async Task PrintAsync(string format, int iterations, int delayMilliseconds)
{
  for (int i = 0; i < iterations; i++)
  {
    await Task.Delay(delayMilliseconds);
    Console.WriteLine(string.Format(format, i));
  }
}
...
await PrintAsync("Iteration {0}", 5, 1_000);
```

(Subsequent releases of C# and .NET have seen many improvements around this asynchronous support, including runtime and compiler enhancements, as well as library additions, such as the introduction of the ValueTask and ValueTask<T> types that enable using async and await with fewer allocations.)

With the compiler providing support for iterators and for async methods, a common question that's asked addresses the combination of the two. You can use yield to author synchronous enumerables, and you can use async and await to author

This article discusses:

- New async interfaces
- Under the hood of async iterators
- Understanding "thread-safe" async enumerables

Technologies discussed:

C# 8, .NET Core 3.0

Code download available at:

github.com/dotnet/reactive

asynchronous operations. What about using yield return with async and await to author asynchronous enumerables?

The answer to that question comes in C# 8 and .NET Core 3.0.

A Tour Through Async Enumerables

.NET Core 3.0 introduces the new System.Collections.Generic.IAsyncEnumerable<T> and System.Collections.Generic.IAsyncEnumerator<T> interfaces. These interfaces, shown in **Figure 1**, should look very familiar, as they closely mirror their synchronous generic counterparts, and the concepts map directly: IAsyncEnumerable<T> provides a method to get an enumerator (IAsyncEnumerator<T>), which is disposable and which provides two additional members, one for moving forward to the next element and one for getting the current element. The deviations from the synchronous counterparts should also stand out: "Async" is used as a prefix in the type names and as a suffix in the names of members that may complete asynchronously; GetAsyncEnumerator accepts an optional CancellationToken; MoveNextAsync returns a ValueTask<bool> instead of bool; and DisposeAsync returns a ValueTask instead of void. (You'll also notice the lack of a Reset method, which the synchronous counterpart exposes, but which is effectively deprecated.)

C# provides direct support for async enumerables, just as it does with synchronous enumerables, both for consuming and for producing them. To iterate through them, await foreach is used instead of just foreach:

```
await foreach (int item in RangeAsync(10, 3))
  Console.Write(item + " "); // Prints 10 11 12
```

And, as with the synchronous code, the compiler transforms this into code very much like you'd write manually if using the interfaces directly:

```
IAsyncEnumerator<int> e = RangeAsync(10, 3).GetAsyncEnumerator();
try
{
  while (await e.MoveNextAsync()) Console.Write(e.Current + " ");
}
finally { if (e != null) await e.DisposeAsync(); }
```

To produce an async enumerable, the language supports writing an iterator just as it does for the synchronous case, but with async IAsyncEnumerable<T> instead of IEnumerable<T> in the signature (as with the synchronous support, the language also allows IAsyncEnumerator<T> to be used as the return type instead of IAsyncEnumerable<T>):

```
static async IAsyncEnumerable<int> RangeAsync(int start, int count)
{
  for (int i = 0; i < count; i++)
  {
    await Task.Delay(i);
    yield return start + i;
  }
}
```

The optional CancellationToken argument to GetAsyncEnumerator is used as a way to request cancellation of the enumerator: At any point during the enumeration, if cancellation is requested, an in-progress or subsequent MoveNextAsync call may be interrupted and throw an OperationCanceledException (or some derived type, like a TaskCanceledException). This begs two questions:

- If the token needs to be passed to GetAsyncEnumerator, but it's the compiler that's generating the GetAsyncEnumerator call for my await foreach, how do I pass in a token?
- If the token is passed to GetAsyncEnumerator and it's the compiler that's generating the GetAsyncEnumerator implementation for my async iterator method, from where do I get the passed-in token?

The answer to the first question (which I'll also come back to shortly) is that there's a WithCancellation extension method for IAsyncEnumerable<T>. It accepts a CancellationToken as an argument, and returns a custom struct type that await foreach binds to via a pattern rather than via the IAsyncEnumerable<T> interface, letting you write code like the following:

```
await foreach (int item in RangeAsync(10, 3).WithCancellation(token))
  Console.Write(item + " ");
```

This same pattern-based binding is also used to enable a ConfigureAwait method, which can be chained in a fluent design with WithCancellation, as well:

```
await foreach (int item in RangeAsync(10, 3).WithCancellation(token).
ConfigureAwait(false))
  Console.Write(item + " ");
```

The answer to the second question is a new [EnumeratorCancellation] attribute. You can add a CancellationToken parameter to your async iterator method and annotate it with this attribute. In doing so, the compiler will generate code that will cause the token passed to GetAsyncEnumerator to be visible to the body of the async iterator as that argument:

```
static async IAsyncEnumerable<int> RangeAsync(
  int start, int count,
  [EnumeratorCancellation] CancellationToken cancellationToken = default)
{
  for (int i = 0; i < count; i++)
  {
    await Task.Delay(i, cancellationToken);
    yield return start + i;
  }
}
```

This means that if you write this code:

```
var cts = new CancellationTokenSource();
await foreach (int item in RangeAsync(10, 3).WithCancellation(cts.Token) { ... }
```

the code inside of RangeAsync will see its cancellationToken parameter equal to cts.Token. Of course, because the token is a normal parameter to the iterator method, it's also possible to pass the token directly as an argument:

```
var cts = new CancellationTokenSource();
await foreach (int item in RangeAsync(10, 3, cts.Token) { ... }
```

Figure 1 **New Async Interfaces**

```
namespace System.Collections.Generic
{
  public interface IAsyncEnumerable<out T>
  {
    IAsyncEnumerator<T> GetAsyncEnumerator(
      CancellationToken cancellationToken = default);
  }

  public interface IAsyncEnumerator<out T> : IAsyncDisposable
  {
    ValueTask<bool> MoveNextAsync();
    T Current { get; }
  }
}

namespace System
{
  public interface IAsyncDisposable
  {
    ValueTask DisposeAsync();
  }
}
```

in which case the body of the async iterator will similarly see cts.Token as its cancellationToken. Why two different ways to do it? Passing the token directly to the method is easier, but it doesn't work when you're handed an arbitrary IAsyncEnumerable<T> from some other source but still want to be able to request cancellation of everything that composes it. In corner-cases, it can also be advantageous to pass the token to GetAsyncEnumerator, as doing so avoids "burning in" the token in the case where the single enumerable will be enumerated multiple times: By passing it to GetAsyncEnumerator, a different token can be passed each time. Of course, it's also possible that two different tokens end up getting passed into the same iterator, one as an argument to the iterator and one via GetAsyncEnumerator. In that case, the compiler-generated code handles this by creating a new linked token that will have cancellation requested when either of the two tokens has cancellation requested, and that new "combined" token will be the one the iterator body sees.

Under the Hood of Async Iterators

Async iterators are transformed by the C# compiler into a state machine, enabling the developer to write simple code while the compiler handles all the complicated intricacies to provide an efficient implementation.

Let's consider the RangeAsync method shown earlier. To begin, the compiler emits the method the developer wrote, but with the body replaced by code that sets up and returns the enumerable object:

```
[AsyncIteratorStateMachine(typeof(<RangeAsync>d__1))]
static IAsyncEnumerable<int> RangeAsync(int start, int count,
  [EnumeratorCancellation] CancellationToken cancellationToken = default) =>
  new <RangeAsync>d__1(-2) {
    <>3__start = start,
    <>3__count = count,
    <>3__cancellationToken = cancellationToken
  };
```

You can see that the method has the same signature as was written by the developer (except for the async keyword, which, as with the async keyword on async methods prior to C# 8, only affects how the compiler compiles the method and doesn't actually impact the method's signature in metadata), and that its sole purpose is to initialize a new instance of the <RangeAsync>d__1 type, which is the compiler-generated IAsyncEnumerable<int> implementation outlined in **Figure 2**. For such a "simple" method as the developer wrote in RangeAsync, there's a lot going on here, so I'll break it into pieces.

First, note that this type not only implements IAsyncEnumerable<int>, but also a bunch of other interfaces. This isn't necessary from a functionality perspective, but it's critical from a performance standpoint. The compiler implementation has been designed to keep allocations incredibly low; in fact, no matter how many times an async iterator yields, the most common case is that it incurs at most two allocations of overhead. To start, this is achieved by employing the same trick that synchronous iterators employ: The same object that implements the enumerable is reused as the enumerator as long as no one else is currently using it. That's why the object also implements IAsyncEnumerator<int>, because the same object typically doubles as both, returning itself from its GetAsyncEnumerator method.

Then there's the IAsyncStateMachine interface. The supporting APIs in the core libraries and runtime operate over abstract state machines as represented by this interface, so in order to be able to perform awaits, the class must implement this interface (it could employ a helper type to implement the interface, but that would be more allocation).

Arguably the most interesting interfaces, however, are IValueTaskSource<bool> and IValueTaskSource, as this gets at the heart of how async enumerables can have so little overhead. When we first designed the async enumerable interfaces, the MoveNextAsync method returned a Task<bool>. The most common case is for the MoveNextAsync method to actually complete its operation synchronously, in which case the runtime would be able to use a cached task object: Every time it completes synchronously to return a true value, the same already-completed-with-a-true-result Task<bool> could be returned, making the synchronously completing case allocation-free. However, .NET Core 2.1 introduced the ability for a ValueTask<T> to be backed not just by a T or by a Task<T>, but also by a new IValueTaskSource<T> interface. This is powerful in that a developer is able to craft an implementation of IValueTaskSource<T> that can be reset and then reused with subsequent ValueTask<T>s. (For more information, see bit.ly/2kEyo81.) By having MoveNextAsync return a ValueTask<bool> instead of a Task<bool>, the compiler can create such an IValueTaskSource<-

Figure 2 **Compiler-Generated IAsyncEnumerable<T>**

```
[CompilerGenerated]
private sealed class <RangeAsync>d__1 :
  IAsyncEnumerable<int>, IAsyncEnumerator<int>,
  IAsyncStateMachine,
  IValueTaskSource<bool>, IValueTaskSource,
{
  private CancellationTokenSource <>x__combinedTokens;
  public CancellationToken <>3__cancellationToken;
  public int <>3__start;
  public int <>3__count;

  public AsyncIteratorMethodBuilder <>t__builder;
  public ManualResetValueTaskSourceCore<bool> <>v__promiseOfValueOrEnd;

  public int <>1__state;
  private int <>l__initialThreadId;
  private bool <>w__disposeMode;

  private CancellationToken cancellationToken;
  private int start;
  private int count;
  private int <i>5__2;
  private TaskAwaiter <>u__1;
  private int <>2__current;

  public <RangeAsync>d__1(int <>1__state) { ... }

  IAsyncEnumerator<int> IAsyncEnumerable<int>.GetAsyncEnumerator(
    Cancellationb cancellationToken) { ... }

  ValueTask<bool> IAsyncEnumerator<int>.MoveNextAsync() { ... }
  int IAsyncEnumerator<int>.Current { get { ... } }

  ValueTask IAsyncDisposable.DisposeAsync() { ... }

  void IAsyncStateMachine.MoveNext() { ... }

  ValueTaskSourceStatus IValueTaskSource<bool>.GetStatus(short token) { ... }
  ValueTaskSourceStatus IValueTaskSource.GetStatus(short token) { ... }

  void IValueTaskSource<bool>.OnCompleted(Action<object> continuation, object state,
    short token, ValueTaskSourceOnCompletedFlags flags) { ... }
  void IValueTaskSource.OnCompleted(Action<object> continuation, object state,
    short token, ValueTaskSourceOnCompletedFlags flags) { ... }

  bool IValueTaskSource<bool>.GetResult(short token) { ... }
  void IValueTaskSource.GetResult(short token) { ... }
}
```

bool> implementation, with every MoveNextAsync that completes synchronously returning a ValueTask<bool> that just wraps a bool, but every MoveNextAsync that completes asynchronously returning a ValueTask<bool> that wraps one of these reusable IValueTaskSource<bool> implementations. And the compiler-generated async enumerable object not only doubles as the enumerator and not only doubles as the async state machine object, it also doubles as exactly that IValueTaskSource<bool> implementation. The same applies to DisposeAsync. Most implementations of DisposeAsync will actually complete synchronously, in which case it can just return a default ValueTask. But if it needs to complete asynchronously, the compiler-generated implementation will just return a ValueTask wrapped around this same async enumerable object, which implements IValueTaskSource, as well. So, no matter how many times MoveNextAsync needs to complete asynchronously, and whether or not DisposeAsync completes asynchronously, they don't incur any additional overhead of allocations, because they just return this instance wrapped in a ValueTask<bool> or ValueTask, respectively.

After the interfaces, you see a bunch of fields. Some of these, like <>3__start and <>3__count, are there to store the arguments passed to the entry point method (you can see them being set to the arguments in the method shown earlier). These values need to be preserved in case the enumerable is enumerated again, which is why you also see start and count fields; those get initialized to their <>3_ counterparts in the GetAsyncEnumerator method and are then the actual fields manipulated when the corresponding "parameters" in the developer's code in the async iterator are read and written. Other fields, like <i>5__2, represent the "locals" used in the async iterator (in this case, it's the i iteration variable in the for loop); any "local" that needs to survive across an await boundary is "lifted" to the state machine in this fashion. But arguably the two most interesting fields are the <>t__builder and <>v__promiseOfValueOrEnd fields. The former represents the lifetime of the asynchronous execution, and provides the facilities for the async iterator to hook in with the runtime's support for things like ExecutionContext flow (ensuring that, for example, AsyncLocal<T> values are properly flowed across awaits). The latter is a ManualResetValueTaskSourceCore<T>, a type introduced in .NET Core 3.0 to contain most of the logic necessary to properly implement IValueTaskSource<T> and IValueTaskSource; the compiler-generated class implements the interfaces, but then delegates the implementations of these interface methods to this mutable struct stored in its field:

```
bool IValueTaskSource<bool>.GetResult(short token) =>
  <>v__promiseOfValueOrEnd.GetResult(token);

void IValueTaskSource<bool>.OnCompleted(
  Action<object> continuation, object state,
  short token, ValueTaskSourceOnCompletedFlags flags) =>
  <>v__promiseOfValueOrEnd.OnCompleted(continuation, state, token, flags);

ValueTaskSourceStatus IValueTaskSource<bool>.GetStatus(short token) =>
  <>v__promiseOfValueOrEnd.GetStatus(token);
```

The remainder of the implementation is really about the state machine itself and moving it forward. The code the developer writes in the async iterator is moved into a MoveNext helper method (**Figure 3** shows an approximate decompilation of the IL generated by the C# compiler), just as is done for synchronous iterators and async methods. There are three main ways to return out of this void-returning helper: the code yields a current value, the code awaits something that hasn't yet completed, or the code reaches the end of the enumeration (either successfully or via an unhandled exception). When the code yields a value, it stores that value into the <>2__current field and updates the <>1__state to indicate where the state machine should jump back to the next time MoveNext is invoked. When the code awaits an incomplete awaiter, it similarly sets the <>1__state (to the location of the code that checks the result of the then-completed awaited operation) and uses the <>t__builder to hook up a continuation that will cause the implementation to call MoveNext again (at which point it will jump to the location dictated by <>1__state). While the implementation may look complicated, that's effectively

Figure 3 **State Machine Implementation**

```
private void MoveNext()
{
  try
  {
    TaskAwaiter awaiter;
    switch (<>1__state)
    {
      default:
        if (<>w__disposeMode) goto DONE_ITERATING;
        <>1__state = -1;
        <i>5__2 = 0; // int i = 0;
        goto LOOP_CONDITION;

      case 0:
        awaiter = <>u__1;
        <>u__1 = default(TaskAwaiter);
        <>1__state = -1;
        goto DONE_AWAIT;

      case -4:
        <>1__state = -1;
        if (<>w__disposeMode) goto DONE_ITERATING;
        <i>5__2++; // i++
        goto LOOP_CONDITION;
    }

    LOOP_CONDITION:
    // i < count
    if (<i>5__2 >= count) goto DONE_ITERATING;
    awaiter = Task.Delay(<i>5__2, cancellationToken).GetAwaiter();
    if (!awaiter.IsCompleted) // await Task.Delay(i, cancellationToken);
    {
      <>1__state = 0;
      <>u__1 = awaiter;
      <RangeAsync>d__1 sm = this;
      <>t__builder.AwaitUnsafeOnCompleted(ref awaiter, ref sm);
      return;
    }

    DONE_AWAIT:
    awaiter.GetResult();
    <>2__current = start + <i>5__2;
    <>1__state = -4;
    goto RETURN_TRUE_FROM_MOVENEXTASYNC; // yield return start + i;

    DONE_ITERATING:
    <>1__state = -2;
    <>x__combinedTokens.Dispose();
    <>v__promiseOfValueOrEnd.SetResult(result: false);
    return;

    RETURN_TRUE_FROM_MOVENEXTASYNC:
    <>v__promiseOfValueOrEnd.SetResult(result: true);
  }
  catch (Exception e)
  {
    <>1__state = -2;
    <>x__combinedTokens?.Dispose();
    <>v__promiseOfValueOrEnd.SetException(e);
  }
}
```

all it is: the developer's code interspersed with the logic for handling yields and awaits in this manner, plus a "jump table" at the beginning of the method that looks at the <>1__state and decides where to go based on it. Much of the remaining complication comes from error handling, as well as from the ability to execute finally blocks as part of DisposeAsync if the enumerator is disposed before it reaches the end, such as if code breaks out of an await foreach loop early.

Finally, there's the MoveNextAsync method (**Figure 4**), which is comparatively simple. If the enumerator isn't in a good state for MoveNextAsync to be called (for example, it's already been disposed), then it just returns the equivalent of "new ValueTask<bool>(false)." Otherwise, it resets the ManualResetValueTaskSourceCore<bool> for the next iteration and calls (via a runtime helper) the MoveNext method just shown. Then, if the invocation completed synchronously, a Boolean indicating whether it successfully moved next or hit the end of the iteration is returned wrapped in a ValueTask<bool>, and if it's completing asynchronously, this object is wrapped in the returned ValueTask<bool>.

All of this is, of course, implementation detail and could easily change in the future. In fact, there are several additional optimizations the compiler can employ and hopefully will in future releases. The beauty of this is you get to keep writing the simple code you want in your async iterators and the compiler handles the details; and as the compiler improves, so, too, do your libraries and applications. The same goes for the runtime and supporting libraries. For example, .NET Core 2.1 and 3.0 both saw significant improvements in the infrastructure supporting async methods, such that existing async methods just got better, and those improvements accrue to async iterators, as well.

Much Ado About Threading

It can be tempting to think of things that are asynchronous as also being "thread-safe" and then jumping to conclusions based on that, so it's important to understand what is and what is not safe when working with async enumerables.

It should be evident that it's fine for one MoveNextAsync call to occur on a different thread from a previous or subsequent MoveNextAsync call; after all, the implementation may await a task and continue execution somewhere else. However, that doesn't mean MoveNextAsync is "thread-safe"—far from it. On a given async enumerator, MoveNextAsync must never be invoked concurrently, meaning MoveNextAsync shouldn't be called again on a given enumerator until the previous call to it has completed. Similarly, DisposeAsync on an iterator shouldn't be invoked while either MoveNextAsync or DisposeAsync on that same enumerator is still in flight.

These rules are easy to follow, and you'd be hard-pressed not to follow them when using await foreach, which naturally follows the rules as part of the code it translates into. However, it's possible to write slightly different code and find yourself with a problem. Consider this buggy variant:

```
IAsyncEnumerable<int> src = ...;
IAsyncEnumerator<int> e = src.GetAsyncEnumerator();
try
{
  while (await e.MoveNextAsync().TimeoutAfter(30)) // BUG!!
    Use(e.Current);
}
finally { if (e != null) await e.DisposeAsync(); }
```

This snippet is using a hypothetical TimeoutAfter method; it doesn't actually exist in .NET Core 3.0, but imagine that it did or that someone wrote it as an extension method, with the semantics that if the task on which it's called hasn't completed within the specified timeout, it'll throw an exception. Now consider this in the context of the previous rules: If this timeout were hit, that means the MoveNextAsync was still in flight, but the TimeoutAfter would cause the iterator to resume with an exception, the finally block to be entered, and DisposeAsync to be called on the enumerator that may still have MoveNextAsync in progress. This could end up failing in a variety of ways, or it could end up accidentally succeeding; in any event, stay away from code like that.

What About LINQ?

Language Integrated Query, or LINQ, provides both a set of helper methods for operating on synchronous enumerables and a set of keywords in the language for writing queries that then compile down to these helper methods. .NET Core 3.0 and C# 8 don't include either of those for asynchronous enumerables. However, the github.com/dotnet/reactive project includes the System.Linq.Async library, which provides a full set of such extension methods for operating on IAsyncEnumerable<T>. You can include this library from NuGet in your project, and have access to a wide array of helpful extension methods for operating over IAsyncEnumerable<T> objects.

What's Next?

C# 8 and .NET Core 3.0 are exciting releases. They include not only the aforementioned language and library support for async enumerables, but also a variety of types that produce or consume them (for example, the System.Threading.Channels.ChannelReader<T> type provides a ReadAllAsync method that returns an IAsyncEnumerable<T>). However, this really is just the beginning for async enumerables. I expect subsequent releases will see further support in the libraries, improvements in the compiler, and additional language functionality. On top of that, I expect we'll see many NuGet libraries for interacting with IAsyncEnumerable<T>, just as we do for IEnumerable<T>. I'm looking forward to seeing all of the ways this feature set will be put to great use. Enjoy! ■

Figure 4 **MoveNextAsync Method**

```
ValueTask<bool> IAsyncEnumerator<int>.MoveNextAsync()
{
  if (<>1__state == -2)
    return default;

  <>v__promiseOfValueOrEnd.Reset();

  <RangeAsync>d__1 stateMachine = this;
  <>t__builder.MoveNext(ref stateMachine);

  short version = <>v__promiseOfValueOrEnd.Version;
  return <>v__promiseOfValueOrEnd.GetStatus(version) ==
ValueTaskSourceStatus.Succeeded ?
    new ValueTask<bool>(<>v__promiseOfValueOrEnd.GetResult(version)) :
    new ValueTask<bool>(this, version);
}
```

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

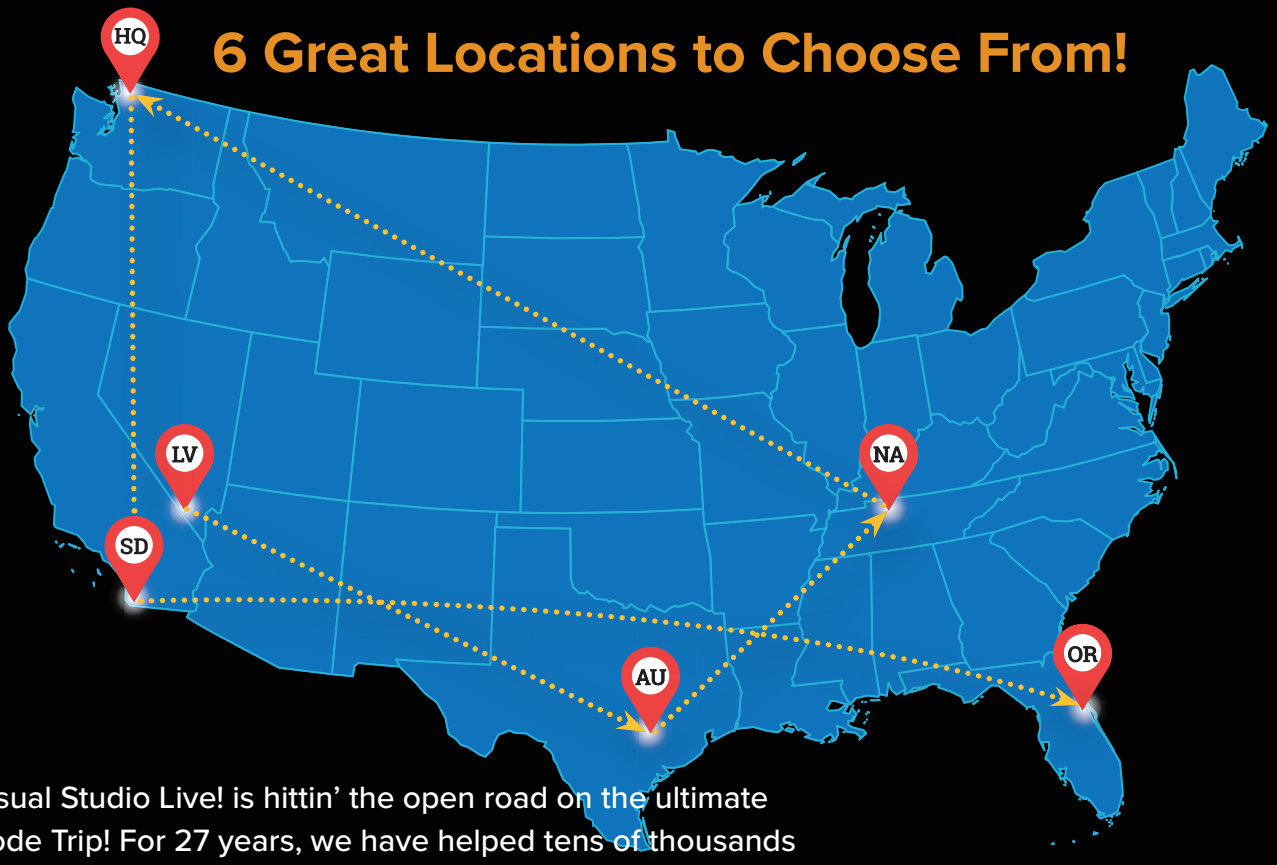
Thanks to the following Microsoft technical experts for reviewing this article: *Julien Couvreur, Jared Parsons*

Visual Studio LIVE!
EXPERT SOLUTIONS FOR ENTERPRISE DEVELOPERS

VSLIVE! 1993 CODE TRIP 2020 A DEVELOPER'S ROAD MAP

Start Your Engines, VSLive!'s 2020 Dates Announced

6 Great Locations to Choose From!



Visual Studio Live! is hittin' the open road on the ultimate Code Trip! For 27 years, we have helped tens of thousands of developers navigate the Developer Highway, featuring code-filled days, networking nights, and the best in independent training:

- Multi-track Events
- Focused, Cutting-edge Microsoft and .NET Core training
- Covering the Hottest Topics
- Relevant and Useable Content
- Expert Speakers, Including Many Microsoft Instructors
- Interactive Sessions, Workshops, & Hands-On Labs
- Discuss Your Challenges
- Find Actionable Solutions

SUPPORTED BY
Microsoft Visual Studio msdn magazine Visual Studio MAGAZINE

PRODUCED BY
CONVERGE 360 An 1105 MEDIA company

Visual Studio Live! is taking the ultimate Code Trip to help you navigate the Developer Highway

JOIN US IN 2020!

March 1-6, 2020

Bally's Hotel & Casino

Visual Studio Live! Las Vegas kicks off it's 2020 Code Trip on The Strip for up to 6-days of developer training on the Microsoft Platform.

REGISTER NOW!
vslive.com/lasvegas

March 30 - April 3, 2020

Hyatt Regency Austin

Visual Studio Live! is pullin' back into Austin in March, where the Texas attitude is big, and the code is bigger!

REGISTER NOW!
vslive.com/austin

May 17 - 21, 2020

Loews Vanderbilt Hotel

Join Visual Studio Live! on its third stop of 2020 as we head to Nashville for the first time in our 26+ year history. Come code by day and experience Music City by night with your fellow peers for in-depth developer training.

REGISTER NOW!
vslive.com/nashville

August 3-7, 2020

Microsoft Conference Center

Redmond, WA

No VSLive! code trip would be complete without a stop where it all began, so we're heading to the idyllic Microsoft Headquarters in Redmond, WA, August 3 – 7, 2020!

REGISTER NOW!
vslive.com/microsofthq

September 27 - October 1, 2020

Hard Rock Hotel San Diego

The VSLive! code trip continues with an in-depth content cruise through Southern California to San Diego!

REGISTER NOW!
vslive.com/sandiego

November 15-20, 2020

Loews Royal Pacific Resort

Visual Studio Live! Orlando is a part of Live! 360, uniquely offering you 6 co-located conferences for one great price! Stay ahead of the current trends and advance your career – join us for the last stop on our 2020 Code Trip!

We Put You In The Drivers Seat with VSLive! On-Demand

Can't get out of the office or want to extend your conference training? Get cutting-edge developer content at home or the office in an easy, flexible format with VSLive! Conference On-Demand! Choose from hundreds of sessions and pick the package that meets your training needs and budget.

Visit ondemand.vslive.com for more information.

Programming Smart Contracts in C#

John deVadoss, Peng Huang

Blockchain platforms have led to incredible advances in the design and development of decentralized applications and systems, and have been applied to domains ranging from cryptocurrencies to enterprise supply chains. *Smart contracts* running on blockchain platforms are the core building block of this new wave of decentralized applications.

Smart contracts provide an execution environment for the economic and transactional logic that encompasses elements of a real-world contract, as well as execution of the terms of the contract. Running on massively scalable, decentralized and immutable blockchain platforms, they have the potential to reshape our economic institutions and the economic relationships and transactions that constitute these institutions.

In this article we guide you through the steps in developing, debugging and deploying your first smart contract using C#.

Some of the technology discussed in this article is in preview. All information is subject to change.

This article discusses:

- Blockchain platforms
- Smart contracts
- Writing a real-world smart contract

Technologies discussed:

C#, .NET Core, NEO Blockchain Toolkit for .NET, Neo Express

What Are Smart Contracts?

The term smart contract was first formalized by cryptographer Nick Szabo at the University of Washington in 1996. According to Szabo, "New institutions, and new ways to formalize the relationships that make up these institutions, are now made possible by the digital revolution …. A smart contract is a set of promises, specified in digital form, including protocols within which the parties perform on these promises."

The easiest way to explain what a smart contract does is through an example. If you've bought a home or you know someone who has, you know there are multiple steps that make up what is often a time-consuming and tedious process. If you're obtaining financing to close your new home, that's another set of steps involving credit agencies, mortgage banks, escrow agencies and more. Each set of workflows and activities is fraught with potential errors, and multiple individuals and commercial entities are often forced to retrace the steps to finally move the buyer into their new home. Along the way, the buyer will have had to interact with a host of individuals, including the salesperson, finance broker, mortgage agent, escrow agent and lender. And, to compensate their work, various fees and commissions are added at each step of the process.

Smart contracts can automate the otherwise confusing and arduous process behind a mortgage contract. A smart contract in this scenario programmatically connects the different parties involved with the mortgage transactions, allowing for a secure, compliant, and scalable process, while making the workflow convenient, frictionless and less error-prone for the individuals.

Does your company create enterprise computing solutions for customers?

If so, does your ability to successfully win clients depend on their understanding of the vastly complex Microsoft stack?

CHALLENGE

The Microsoft ecosystem is moving at a blistering pace

Your customers and prospects need clear guidance on some of the key architecture and directional questions for how to proceed.

SOLUTION

Redmond Intelligence Research Reports

Actionable research and intelligence reports from our network of Microsoft experts, including Most Valuable Professionals (MVPs).

Sponsor a Redmond Intelligence Research Report, a Best Practices guide or a Solution Spotlight report. You'll come away with an authoritative, independent report, professionally edited and designed by the team behind Converge360, which you exclusively distribute. Contact us today for more details.

Redmond INTELLIGENCE
redmondintelligence.com

CONTACT

Dan LaBianca | General Manager | **Voice** 818.674.3416 / **E-mail** dlabianca@Converge360.com

CONVERGE 360
An 1105 MEDIA company

A smart contract holds the economic and transactional logic that includes the elements of an ordinary contract (for example, offer, acceptance, revocation), as well as the execution of the terms (such as payment, price variation, penalties) that comprise the contract. It enables the processing of transactional actions at predetermined times and/or based in relation to the occurrence or non-occurrence of an external trigger, such as an exchange rate threshold, delivery of an asset or expiration of put/call.

The key capabilities enabled using a smart contract involve:

- The ability to authenticate parties and counterparties, ownership of assets and claims of right.
- The ability to access and refer to information and data both on the blockchain platform and outside of the smart contract (and the blockchain) to trigger transactions.
- The ability to automate the execution of transactions and (economic) protocols on the blockchain platform.

What Can Smart Contracts Do?

The potential benefits of smart contracts throughout an economic transaction include standardization (standardized schemas and protocols reducing the cost of negotiations and agreements), security (transactions are encrypted and stored on a blockchain platform designed to be immutable), latency (reduced transaction times and the streamlining or elimination of redundant manual processes) and transaction certainty (programmed execution, which reduces counterparty risk and settlement risk) and more.

As we examine the institutional landscape today, the use cases for smart contracts cover a broad range of applications:

- **Self-Sovereign Identity:** Enable individuals to own and control their digital identity, which contains reputation, data and digital assets.
- **Securities:** Simplify capitalization table management and the circumvention of intermediaries in the chain of securities custody.
- **Trade Finance:** Facilitate streamlined international transfers of goods through faster letters of credit and trade payment initiation, while enabling higher liquidity of financial assets.
- **Data Governance and Compliance:** Enable uniform financial data across organizations and improved financial reporting while reducing auditing and assurance costs.
- **Supply Chains:** Use IoT to track products as they move from the factory floor to the store shelf, enabling real-time visibility across the entire supply chain.
- **Clinical Research and Trials:** Increase cross-institutional visibility and data sharing while preserving privacy.

With a perspective on the potential of smart contracts, let's explore the structure and development of our first smart contract.

The Hello World Smart Contract

Following hallowed computer science traditions, let's consider our first Hello World contract. **Figure 1** shows the code.

Every smart contract inherits the SmartContract base class, which is in the Neo Framework and provides core methods. The Neo namespace surfaces the APIs made available by the NEO blockchain platform, providing a way for the smart contract to access blockchain data (all the data on the entire blockchain, including complete blocks and transactions, as well as each of their fields) and to manipulate the persistent store (every smart contract deployed has a storage space that can only be accessed by the contract itself).

Inside the contract class, properties defined with static readonly or const are contract properties, which can be used as constants. For instance, to define the owner of a contract or the factor number that will be used in subsequent asset transfers, we define them like this:

```
// Represents the owner of this contract, which is a fixed address
public static readonly byte[] Owner =
  "ATrzHaicmhRj15C3Vv6e6gLfLqhSD2PtTr".ToScriptHash();

// A constant factor number
private const ulong factor = 100000000;
```

In addition, you can define static methods in the contract class and return a constant.

When you develop a smart contract, the blockchain platform provides a means to store your application data on the blockchain. All data in a smart contract's storage is automatically persisted across invocations of the smart contract. Full nodes in the blockchain store the state of every smart contract on the chain.

The NEO blockchain platform provides data access interfaces based on key-value pairs. Our first smart contract uses the Storage class to read and write to persistent storage. For instance, to store the total supply of your token, you use this code:

```
// Key is totalSupply and value is 100000000
Storage.Put(Storage.CurrentContext, "totalSupply", 100000000);
```

Here, CurrentContext returns the current storage context. After obtaining the storage context, the object can be passed as an argument to other contracts (as a way of authorization), allowing them to perform read and write operations on the persistent store of the current contract.

With that, let's explore a simple real-world scenario.

A Real-World Smart Contract

Consider a simplified DNS scenario where we want to register, query and delete a domain name associated with a given user, as shown in **Figure 2**.

In theory, smart contracts can have any entry point, but usually the Main function serves as the entry point for ease of invocation.

Triggers A smart contract trigger is a mechanism that activates the execution of a smart contract. The most commonly used triggers are verification triggers and application triggers. Typically, you handle the triggers in the Main function.

Figure 1 **The Hello World Smart Contract**

```
using Neo.SmartContract.Framework;
using Neo.SmartContract.Framework.Services.Neo;
using System;

namespace HelloWorld
{
  public class HelloWorld : SmartContract
  {
    private const string test_str = "Hello World";
    public static String Main(string operation, object[] args)
    {
      Storage.Put("Hello", "World");
      return test_str;
    }
  }
}
```

STEAM UNIVERSE
THE WAY FORWARD FOR EDUCATION

IN PARTNERSHIP WITH
THE JOURNAL
CAMPUS TECHNOLOGY

YOUR NEW GO-TO RESOURCE

STEAM FOR EDUCATION

- Artificial Intelligence
- Makerspaces
- Computer Science
- Grants
- Professional Development
- 3D Printers
- Internet of Things

And MORE!

STEAMUniverse.com

A verification trigger is used to invoke the contract as a verification function, accepting multiple parameters and returning a valid Boolean value, indicating the validity of the transaction or block. The contract code is executed to verify whether a transaction involving assets owned by the contract address should be allowed to succeed.

When you transfer assets from account A to account B, verification is triggered. All the nodes in the blockchain that received the transaction verify account A's contract. If the return value is true, the transfer completes successfully. If the return value is false, the transfer will fail and the transaction won't be recorded in the blockchain:

```
public static bool Main(byte[] signature)
{
  if (Runtime.Trigger == TriggerType.Verification)
  {
    if (/*condition A*/)
        return true;
      else
        return false;
    }
}
```

An application trigger is used to invoke the contract and you can expect the input arguments with the types you specified to be present. You'll utilize these input values to determine the resulting state of the contract, and its output.

Unlike the verification trigger, which is activated by a transfer, an application trigger (shown in **Figure 3**) is activated by a special transaction, InvocationTransaction. Because the contract is executed after InvocationTransaction is confirmed, the transaction is recorded in the blockchain irrespective of whether the smart contract execution succeeds or fails.

In this example, because it's a smart contract without asset transfers involved, the only trigger that needs to be considered is the application trigger.

CheckWitness In many, if not all, cases, you'll want to validate whether the address invoking your contract code is really who it says it is. The Runtime.CheckWitness method accepts a single parameter that represents the address you'd like to validate against the address used to invoke the contract code. More specifically, it verifies that the transactions or block of the calling contract has validated the required script hashes.

Programming Your First C# Smart Contract

We'll use the preview release of the NEO Blockchain Toolkit for .NET on Visual Studio Code Marketplace to develop, debug and deploy our first smart contract using C#. For brevity, rather than using the Visual Studio Code extensions, we'll use command-line tools to illustrate the steps. We'll use the Hello World contract as our example.

In your terminal window, create an empty directory called HelloWorld. Change to that directory and invoke the dotnet new new-contract command. This will create a NEO smart contract that writes Hello World to blockchain storage. If you wish, you can create a Visual Basic smart contract by adding -lang VB to the command prior to execution.

You can immediately build the smart contract via the dotnet build command. The result should look something like **Figure 4**.

From the terminal window in your HelloWorld project directory, you can launch Visual Studio Code by executing code.

Before you can run the contract in the debugger, you need to create a launch configuration. The NEO smart contract debugger makes this very easy. From the top-level Debug menu select Add Configuration, and then from the Select Environment input box select NEO Contract.

When you hit F5 or select Debug | Start Debugging from the menu, the HelloWorld contract launches for debugging. At this point, you can do any of the following:

- Continue, Step Into, Step Over and Step In
- Set Breakpoints

Figure 2 **DNS Smart Contract**

```
using Neo.SmartContract.Framework;
using Neo.SmartContract.Framework.Services.Neo;
namespace Neo.SmartContract
{
  public class Domain : SmartContract
  {
    public static object Main(string operation, params object[] args)
    {
      if (Runtime.Trigger == TriggerType.Application){
            switch (operation){
              case "query":
                return Query((string)args[0]);
              case "register":
                return Register((string)args[0], (byte[])args[1]);
              case "delete":
                return Delete((string)args[0]);
              default:
                return false;
            }
      }
    }

    private static byte[] Query(string domain)
    {
      return Storage.Get(Storage.CurrentContext, domain);
    }


    private static bool Register(string domain, byte[] owner)
    {
      // Check if  the owner is the same as the one who invoke the contract
      if (!Runtime.CheckWitness(owner)) return false;
      byte[] value = Storage.Get(Storage.CurrentContext, domain);
      if (value != null) return false;
      Storage.Put(Storage.CurrentContext, domain, owner);
      return true;
    }

    private static bool Delete(string domain)
    {
      // To do
    }
  }
}
```

Figure 3 **Application Trigger**

```
public static Object Main(string operation, params object[] args)
{
  if (Runtime.Trigger == TriggerType.Verification)
  {
    if (/*Condition A*/)
        return true;
      else
        return false;
  }
  if (Runtime.Trigger == TriggerType.Application)
  {
    if (operation == "FunctionA") return FunctionA(args);
  }
}

// There is a smart contract entry point and redirected from main method
public static bool FunctionA(params object[] args)
{
  // Some code
}
```

- Inspect the Contents of emulated storage
- Inspect the Value of local parameters and variables

In order to deploy the smart contract, you need a PrivateNet instance of the blockchain. NEO Express is a developer-focused private network that's built on the MainNet code base and provides full symmetry with the MainNet. We'll use NEO Express to illustrate the steps involved in deployment.

By default, NEO Express creates a single-node blockchain.

Create a new NEO Express instance with the create command. You'll see something like the output in **Figure 5**.

By default, NEO Express creates a single-node blockchain, but you can create a four- or seven-node blockchain with the --count option. Once you've created the NEO Express blockchain instance, you can run it. Because this is a single-node blockchain, you don't need to specify which node of the blockchain to run. You can control the block generation period via the --seconds-per-block option (-s for short) of the run command.

Because this terminal window is running the blockchain, open another terminal window in the same directory so you can interact with the running PrivateNet. In the new terminal window, you'll create a standard wallet and transfer the "genesis" NEO tokens to that wallet, as you'll need GAS to be able to deploy and invoke the contract, as shown in **Figure 6**.

You can see how much GAS is available with the show gas command. In order to access the GAS, you need to execute another transfer, this time from and to the testWallet account. The available GAS can be claimed with the claim gas command and you can see the result with the show account command.

With a running NEO Express blockchain and a standard wallet account with plenty of GAS, you can deploy the smart contract to the blockchain. You start by importing the contract into NEO Express. NEO Express needs to know if the contract uses storage, dynamic invoke or if the contract is payable in order to deploy the contract. For NEO 3, this information will be in the smart contract manifest file.

Figure 4 **Building the Smart Contract**

```
$ dotnet build
Microsoft (R) Build Engine version 16.2.32702+c4012a063 for .NET Core
Copyright (C) Microsoft Corporation. All rights reserved.

  Restore completed in 294.77 ms for /home/jdevados/Source/HelloWorld/
HelloWorld.csproj.
  HelloWorld -> /home/jdevados/Source/HelloWorld/bin/Debug/netstandard2.0/
HelloWorld.dll
  HelloWorld -> /home/jdevados/Source/HelloWorld/bin/Debug/netstandard2.0/publish/
  Neo.Compiler.MSIL console app v2.4.1.1
  Find entrypoint:System.Void HelloWorld::Main()
  convert succ
  gen debug succ
  gen md succ
  gen abi succ
  write:HelloWorld.avm
  write:HelloWorld.debug.json
  write:HelloWorld.abi.json
  SUCC
```

Figure 5 **Creating and Running the PrivateNet**

```
$ neo-express create
Created 1 node privatenet at /home/jdevados/Source/HelloWorld/default.neo-express.json
  Note: The private keys for the accounts in this file are are *not* encrypted.
    Do not use these accounts on MainNet or in any other system where
security is a concern.

$ neo-express run --seconds-per-block 1
09:49:37.99 ConsensusService Info OnStart
09:49:38.08 ConsensusService Info initialize: height=1 view=0 index=0 role=Primary
09:49:38.15 ConsensusService Info timeout: height=1 view=0
09:49:38.15 ConsensusService Info send prepare request: height=1 view=0
09:49:38.23 ConsensusService Info send commit
09:49:38.32 ConsensusService Info relay block: height=1
hash=0x096aaa25191b8601856a0a1744b7f19b06807fd9888e247366eec3d212a507b6 tx=1
09:49:41.32 ConsensusService Info persist block: height=1
hash=0x096aaa25191b8601856a0a1744b7f19b06807fd9888e247366eec3d212a507b6 tx=1
09:49:41.32 ConsensusService Info initialize: height=2 view=0 index=0 role=Primary
09:49:42.33 ConsensusService Info timeout: height=2 view=0
...
```

Figure 6 **Deploying the Smart Contract on the PrivateNet**

```
$ neo-express wallet create testWallet
...

$ neo-express transfer neo 100000000 genesis testWallet
...

$ neo-express show account testWallet
{
  ...
  "balances": [
    {
      "asset":
"0xc56f33fc6ecfcd0c225c4ab356fee59390af8560be0e930faebe74a6daff7c9b",
      "value": "100000000"
    }
  ]
}

$ neo-express show gas testWallet
{
  "unavailable": 1112,
  "available": 0
}

$ neo-express transfer neo 100000000 testWallet testWallet
...

$ neo-express show gas testWallet
{
  "unavailable": 96,
  "available": 1408
}

$ neo-express claim gas testWallet
...

$ neo-express show account testWallet
{
  ...
  "balances": [
    {
      "asset":
"0xc56f33fc6ecfcd0c225c4ab356fee59390af8560be0e930faebe74a6daff7c9b",
      "value": "100000000"
    },
    {
      "asset":
"0x602c79718b16e442de58778e148d0b1084e3b2dffd5de6b7b16cee7969282de7",
      "value": "1408"
    }
  ]
}

$ neo-express contract import bin/Debug/netstandard2.0/publish/
Does this contract use storage? [y/N] y
Does this contract use dynamic invoke? [y/N] n
Is this contract payable? [y/N] n

$ neo-express contract deploy HelloWorld testWallet
...
```

Figure 7 **Invoking the Smart Contract on the PrivateNet**

```
$ neo-express contract invoke HelloWorld --account testWallet
{
  "contract-context": < omitted for clarity >
  "script-hashes": < omitted for clarity >
  "hash-data": < omitted for clarity >
  "engine-state": {
    "state": 1,
    "gas-consumed": "1.017",
    "result-stack": []
  }
}
{
  "txid":
    "0x785346a3a338d70dd5bee6a70e1fc807a891d23a8d12d138b6a151b5eeae771e"
}

$ neo-express contract storage helloWorld
0x48656c6c6f
  key (as string)   : Hello
  value (as bytes)  : 0x576f726c64
        (as string) : World
  constant value    : False
```

The imported contract can now be deployed via the contract deploy command. You must specify a wallet account to pay the deployment GAS price. Invoking a smart contract without specifying a wallet account to pay the invocation GAS cost won't modify the state of the blockchain. For the HelloWorld contract, this means that nothing will be written to blockchain storage. If you want a contract invocation to make durable changes, you can specify a wallet account to pay the GAS cost via the --account argument, as shown in **Figure 7**.

The HelloWorld contract takes no parameters and returns no values. However, it does modify contract storage in the blockchain. You can dump the storage state for a given contract with the contract storage command. This command lists all the key/value pairs in the blockchain, showing both keys and values as both a hex-encoded byte array and as a UTF-8 encoded string.

Wrapping Up

Smart contracts promise to transform our economic institutions, and the economic relationships and transactions that constitute these institutions. With benefits including standardization, security, reduced latency and transaction certainty and more, the benefits of smart contracts span domains ranging from the securities market, through clinical research and trials, to enterprise supply chains.

In this article we've focused on the what, why and how of smart contracts and provided a step-by-step approach to developing, debugging and deploying your first smart contract in C#. ■

John deVadoss *leads development for NEO Global Development in Seattle. Previously he built Microsoft Digital, .NET Patterns & Practices and .NET Architecture Strategy. He also incubated Microsoft Azure when he was at Microsoft. Most recently, he launched two machine learning start-ups. deVadoss did his Ph.D. work in machine learning, specializing in recurrent neural networks.*

Peng Huang *is a leading industry blockchain platform strategist. Previously at Microsoft he led the Developer Tools and ISV strategy for Greater China, and product management for .NET Patterns & Practices in Redmond. He's a senior member of ACM and began his career as an Assembly language coder on the SPARC and x86 platforms.*

Thanks to the following technical expert for reviewing this article: *Harry Pierson*

msdn magazine

STATEMENT OF OWNERSHIP, MANAGEMENT AND CIRCULATION

1. Publication Title: MSDN Magazine
2. Publication Number: 1528-4859
3. Filing Date: September 30, 2019
4. Frequency of Issue: Monthly with a special issue in December
5. Number of Issues Published Annually: 13
6. Annual Subscription Price: US $35, International $60
7. Complete Mailing Address of Known Office of Publication: 6300 Canoga Ave., Ste. 1150, Woodland Hills, CA 91367
8. Complete Mailing Address of the Headquarters of General Business Offices of the Publisher: Same as above.
9. Full Name and Complete Mailing Address of Publisher, Editor, and Managing Editor: Dan LaBianca, Chief Revenue Officer, 14901 Quorum Drive, Ste 425, Dallas, TX 75254
   Michael Desmond, Editor-in-Chief, 8251 Greensboro Drive, Suite 510, McLean, VA 22102
10. Owner(s): 1105 Media, Inc. dba: 101 Communications LLC, 6300 Canoga Ave., Ste. 1150, Woodland Hills, CA 91367 Listing of shareholders in 1105 Media, Inc.
11. Known Bondholders, Mortgagees, and Other Security Holders Owning or Holding 1 Percent or more of the Total Amount of Bonds, Mortgages or Other Securities:
    Nautic Partners V, L.P., 50 Kennedy Plaza, 12th Flr., Providence, RI 02903
    Kennedy Plaza Partners III, LLC, 50 Kennedy Plaza, 12th Flr., Providence, RI 02903
    Alta Communications IX, L.P., 1000 Winter Street, South Entrance, Suite 3500, Waltham, MA 02451
    Alta Communications IX, B-L.P., 1000 Winter Street, South Entrance, Suite 3500, Waltham, MA 02451
    Alta Communications IX, Associates LLC, 1000 Winter Street, South Entrance, Suite 3500, Waltham, MA 02451
12. The tax status has not changed during the preceding 12 months.
13. Publication Title: MSDN Magazine
14. Issue date for Circulation Data Below: September 2019
15. Extent & Nature of Circulation:

	Average No. Copies Each Month During Preceding 12 Months	No. Copies of Single Issue Published Nearest to Filing Date
a. Total Number of Copies (Net Press Run)	69,855	64,950
b. Legitimate Paid/and or Requested Distribution		
1. Outside County Paid/Requested Mail Subscriptions Stated on PS Form 3541	47,065	34,508
2. In-County Paid/Requested Mail Subscriptions Stated on PS Form 3541	0	0
3. Sales Through Dealers and Carriers, Street Vendors, Counter Sales, and Other Paid or Requested Distribution Outside USPS®	755	667
4. Requested Copies Distributed by Other Mail Classes Through the USPS®	0	0
c. Total Paid and/or Requested Circulation	47,820	35,175
Nonrequested Distribution		
1. Outside County Nonrequested Copies Stated on PS Form 3541	21,496	29,526
2. In-County Nonrequested Copies Distribution Stated on PS Form 3541	0	0
3. Nonrequested Copies Distribution Through the USPS by Other Classes of Mail	0	0
4. Nonrequested Copies Distributed Outside the Mail	461	205
e. Total Nonrequested Distribution	21,957	29,731
f. Total Distribution	69,777	64,906
g. Copies not Distributed	78	44
h. Total	69,855	64,950
i. Percent paid and/or Requested Circulation	68.53%	54.19%

16. Electronic Copy Circulation
    a. Requested and Paid Electronic Copies
    b. Total Requested and Paid Print Copies (Line 15c) + Requested/Paid Electronic Copies
    c. Total Requested Copy Distribution (Line15f) + Requested/Paid Electronic Copies (Line 16a)
    d. Percent Paid and/or Requested Circulation (Both print & Electronic Copies) (16b divided by 16c x 100)
    ☑ **I certify that 50% of all my distributed copies (electronic and paid print are legitimate request or paid copies.**
17. Publication of Statement of Ownership for a Requester Publication is required and will be printed in the November 2019 issue of this publication.
18. I certify that all information furnished on this form is true and complete:
    Tillie Carlin, Manager, Print Production

Visual Studio MAGAZINE

READER'S CHOICE AWARD WINNER 2020

DevExpress®

Visual Studio MAGAZINE 1 READER'S 2020 CHOICE

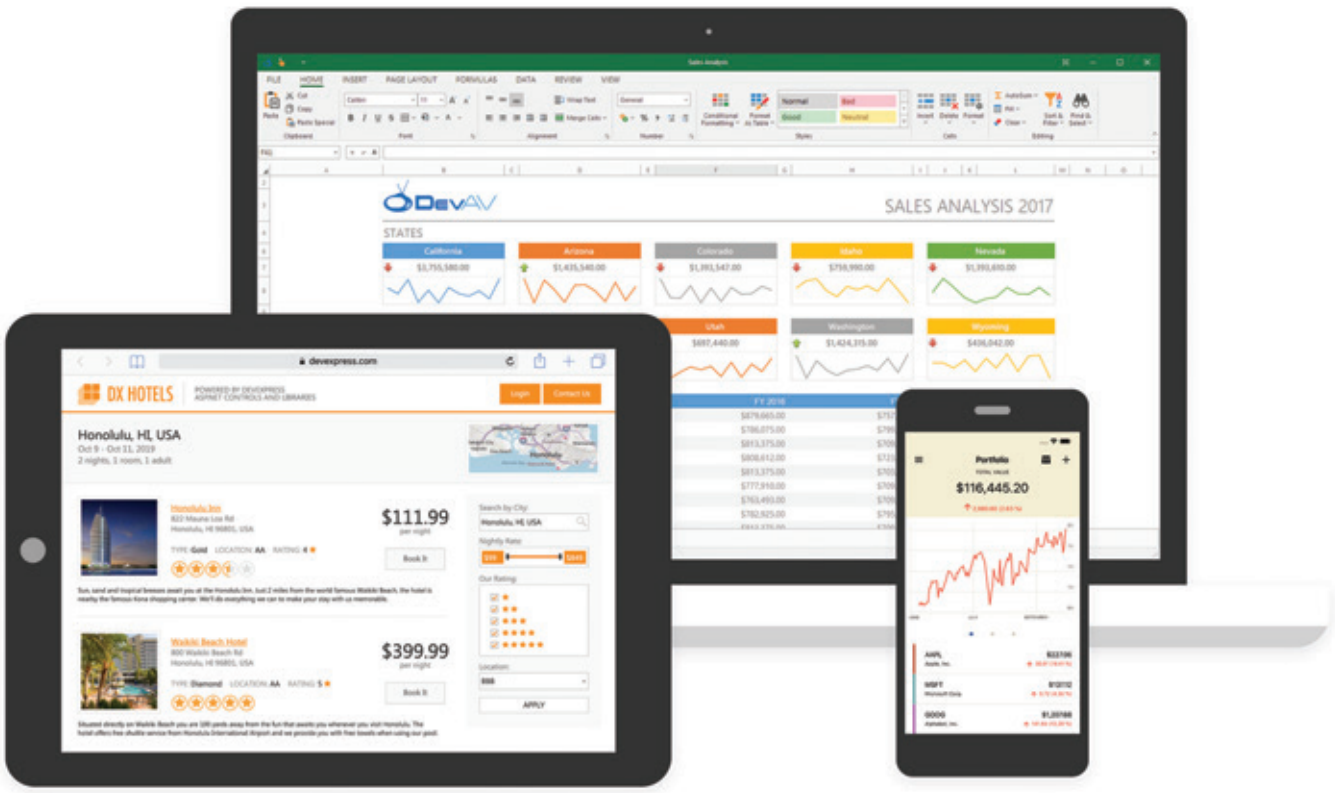
Thank You x 18

We are grateful to everyone who voted for our products in this year's VSM Readers' Choice Awards. Thank you for your continued support and for the confidence you've placed in DevExpress for over 20+ years.

Since 1998, our goal has been steadfast and unshakeable: Create best-of-breed software development tools so you can deliver high-impact business solutions that amaze. From the desktop and web to your mobile world, we remain fully committed to your needs and will do everything possible to earn your trust in the years to come.

Thank you once again from all of us at DevExpress.
management@devexpress.com

MVC WPF UWP .NET CORE BLZ JS



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

3 Things: A Few Last Words on Software

For more than two decades, I've worked to stay true to the mission of this column and to be constantly on the cutting edge of development. Now that this endeavor is coming to its end, I hope to leave you with insight that can help keep you on the cutting edge in the months and years to come. I won't be covering any of the new amazing features coming to ASP.NET Core 3, .NET 5, Blazor, ML.NET or gRPC. Nor will I hazard any guesses as to what the future may bring to the IT and software development world in the next five to 10 years. I'm old enough to heed Socrates' ageless motto: "One thing only I know, and that is that I know nothing."

So in this, my final column, I'll share my perspective about three topics and their related buzzwords that are of crucial relevance today.

The Case for Agile in Artificial Intelligence

Everyone in the industry is using artificial intelligence (AI) as a bullhorn to reach and impress the widest possible audience, and emphasizing AI as a tool that can solve any IT issue and streamline any convoluted process. This is all marketing hype.

At its core, AI isn't magic—it's data crunching, elbow grease and software acumen. Companies don't need AI, per se, they need end-to-end solutions to their concrete business problems and measurable results for their efforts. AI, and more specifically machine learning, is the most powerful tool we have to provide a new generation of end-to-end, tailormade solutions, at a reasonable cost.

The problem is, doing AI in the real world is much more frustrating than people think. For complex real-world problems (say, production forecasts in renewable energy) you don't end up using a canonically trained model. Instead, you work with software artifacts that are only minimally trained (on a small number of variables) and analytically process frequently changing live data such as weather forecasts.

Another great example is fault prediction or predictive maintenance. If you search around, it may seem a problem good for an exercise—you can find plenty of short articles about it on Web sites. The thing is, fault prediction requires a Long Short Term Memory neural network, which is one of the trickiest flavors of neural networks to work with. Still, companies are actively looking into solving the business problem of reducing downtime and optimizing maintenance with more analytical, half-human, half-machine forms of predictions.

All of this is to say that AI is really just a newer and fancier name for problem solving, just with more advanced tools.

In companies, the data science team is often seen as the panacea team and charged with too many (or too few) responsibilities. The reality is that too often a waterfall-style model is established to rule the collaboration between data science and software. One team delivers the trained model; another team mounts it into the host application. Sometimes it works with the live data of the real world, but, more often, it doesn't. I've seen algorithms needing input that no client application could realistically provide. I've seen algorithms trained on data taken from one wind farm and tested for acceptance using data from another wind farm using different turbines and subject to different orography. The trained algorithm needs to be an integrated part of the software solution.

For serious AI, you need an agile process that positions the data science and software teams to work as a single team, or at least to work together closely. The importance of data/software collaboration is the reason I suggest looking into ML.NET rather than committing to Python for general machine learning development. The rich ecosystem of coded algorithms and libraries for data manipulation in ML.NET mean that developers can leverage the power of SQL Server, Entity Framework and the .NET Framework to work with data. There's no reason for .NET shops to use Python—thinking so is a clear sign that you may be missing some crucial points about AI.

Event Sourcing over Blockchain

More than a decade ago, blockchain was devised to function in the context of a very specific scenario: solving the double spending (DS) problem in a deliberately decentralized system. In a nutshell, in a digital cash system, DS is the problem of preventing digital money (which, ultimately, are just files) from being duplicated or tampered with. A physical coin can be spent once at a time, and a person can only spend it only if he or she has legitimately received the coin from another individual. Blockchain is the software platform devised to avoid DS in the context of the Bitcoin cryptocurrency. (By the way, DS is most decidedly not a trivial problem to solve.)

As Bitcoin gained traction, the term blockchain gained ascendancy. But from a development perspective, the various cloud services offering blockchain networks are ultimately providing storage. So what's the difference between blockchain and other storage platforms such as relational databases? One is decentralized governance and another is stronger resistance to tampering. A third aspect is the event-based nature of the blockchain storage.

My personal thinking is that the inherent ability of a blockchain network to track every single transaction is the aspect that makes it so attractive and intriguing. Putting something on the blockchain records that a thing happened, and when and how that thing

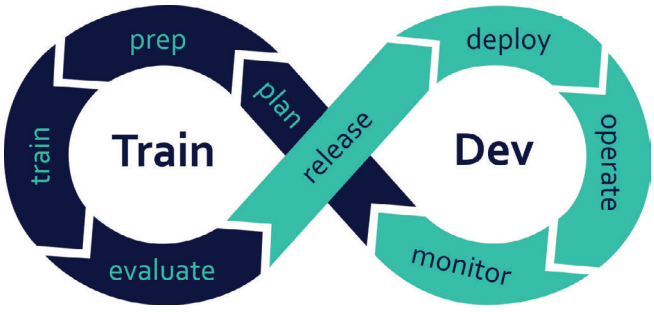


Figure 1 **The Tight Coupling of Software and Data in Artificial Intelligence**

happened, can be easily tracked back. It's as if the coin in your pocket could let you inspect the entire list of handovers it was subject to since it was released from the mint. That's what blockchain was made for—maintaining the history of transactions for all individual Bitcoins with the certainty that no transaction or past facts could be altered. Therefore, blockchain makes perfect sense in the context of Bitcoin and cryptocurrencies.

So in what way can blockchain help (some might say disrupt) the general industry?

The only reasonable value I see is for countrywide, or even transnational, networks that store shared and continuously edited information. Who manages these networks, though? In the end, it's always about storing data at some location (centralized or not) and having someone review and approve it.

The issue goes far beyond the software aspect of storage. If we only talk about traceability of business facts, then event sourcing is a much more concrete alternative. Event sourcing is an approach to building the data layer of software applications in which all the changes to the application state are stored as a sequence of events and each change is treated individually. You have a record (or an event) when an order is created, another event when the order is updated the first time, yet another when it's updated the second time, and so forth. To get the current state of the order, you must replay the entire list of events, similar to how you determine the balance of a bank account. It's a number, but it results from the entire list of operations you made over time.

Event sourcing frameworks work in an append-only mode, with the deletion of objects tracked by adding an event that describes when and why the order was deleted. These frameworks typically don't fully support blockchain key features such as decentralization, voting and anti-tampering measures. But the idea of tracking everything also can be achieved via event sourcing, and from event sourcing with some extensions you can get very close to blockchain. Hence, the focus returns on the actual problem you want to solve rather than how you'd like to solve it.

Over Engineering: Just Say No

We've seen modern software giants court complexity to adapt to rapid growth. Facebook adopted polyglot persistence to serve its hundreds of millions of active users. Amazon switched to Web services, Netflix to microservices and Google to gRPC over REST. Alas, I fear we may be learning the wrong lessons from these success stories.

For instance, today it seems almost inevitable that any Web application must be designed around microservices. I'm old enough to have worked with Distributed COM and I recall the first (and only) law of Distributed Programming: Do Not Distribute It. A distributed architecture has pros and cons and it's never obvious when the pros will overtake the cons. Working within a network brings new issues such as latency and synchronization, and amplifies others such as debugging and security. In a nutshell, everything is more complex.

Complexity is justifiable when it brings a tangible return. But in my experience, starting out with a complex design because it was used successfully in another scenario has never proved wise. Instead, you should constantly monitor the effectiveness of your design and add complexity only where and when necessary. Over-engineering is always costly and risky.

A Farewell Note

I've been asked many times how I can keep up with quickly changing software technologies and grasp the issues related to development problems without writing enterprise code around the clock. My secret is I try to get at the foundation of things and think of software as the mirror of business processes designed by humans. Deep understanding of the business smooths the task of writing software and makes it kind of mechanical work. Deep understanding of technologies makes it easy to see if it fits in a specific business problem.

To a certain extent, software is a physical thing and is subject to familiar physical laws, such as entropy. At the same time, software is a human thing and, as such, is influenced by some context-specific form of empathy—the ability to understand. While we all know about entropy and its impact on code over time, I'm more interested in empathy, which expresses the ability of software to penetrate the internal mechanics of processes. Empathy impacts the quality of software and can even reduce entropy in the long run.

The power of understanding things—call it software empathy—is what I leverage in my every day activities and that I tried to illustrate by example in my columns. I ended up writing because I had always wanted to be a writer, and I found that being a coder was one way to be a writer. So I put passion and empathy into it, the same way I worked to push passion and empathy into my code. Did it work? The fact that this column has thrived for more than 20 years is a clue!

I wish to end with wisdom from some great men, just adapted to software. Let's call them my messages for posterity:

- *Keep it simple, as simple as possible. Otherwise it's over-engineered.* (Adapted from Albert Einstein)
- *Don't forget to pay your technical debt.* (Adapted from the reported last words of Socrates)
- *Perhaps a problem did not want to be solved so much as to be understood.* (Adapted from George Orwell)

Stay in touch at youbiquitous.net. ■

Dino Esposito *has authored more than 20 books and 1,000-plus articles in his 25-year career. Author of "The Sabbatical Break," a theatrical-style show, Esposito is busy writing software for a greener world as the digital strategist at BaxEnergy. Follow him on Twitter: @despos.*

Visual Studio LIVE!
EXPERT SOLUTIONS FOR ENTERPRISE DEVELOPERS

LAS VEGAS CODE TRIP 1993 2020
A DEVELOPER'S ROAD MAP

Join us for six days of unbiased, cutting-edge education on the Microsoft Platform

LAS VEGAS
CODE ON THE STRIP

March 1-6, 2020
Bally's Hotel & Casino

CODE ON THE STRIP

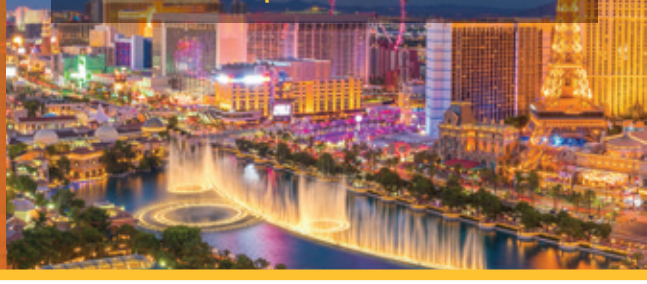
Visual Studio Live!'s first stop on its 2020 Code Trip is Las Vegas, situated fittingly near Historic Route 66. Developers, software architects, engineers, and designers will cruise onto the Strip for six days of unbiased and cutting-edge education on the Microsoft Platform. Let VSLive! be your road map to navigate the .NET Highway with industry experts and Microsoft insiders in 60+ sessions and fun networking events – all designed to keep you and your business competitive and future-ready.

Choose VSLive! Las Vegas:

- To improve your technical skills and become a more valuable asset to the company.
- To learn from both industry experts and Microsoft product team members in a dev-focused environment.
- To become more efficient and productive by shaving time and endless cycles off your development projects.
- To network and build new relationships with other development professionals in the industry.

Don't gamble and miss out on $500 in savings!!

Register by December 20 and save $500.



SUPPORTED BY

Microsoft | Visual Studio | Visual Studio MAGAZINE

PRODUCED BY

CONVERGE 360 an 1105 MEDIA company

#VSLive

AGENDA SNEAK-PEAK – Beat the Traffic to Stay Ahead

Get up to speed on what's happening on the .NET developer landscape now, and learn what's coming next to get out ahead of the competition.

DevOps in the Spotlight	Cloud, Containers and Microservices	AI, Data and Machine Learning	Developing New Experiences	Delivery and Deployment	.NET Core and More	Full Stack Web Development

START TIME	END TIME				
Full Day Hands-On Labs: Sunday, March 1, 2020 *(Separate entry fee required)*					
9:00 AM	6:00 PM	**HOL01** Full Day Hands-On Lab: DevOps with GitHub - *Brian Randell & MickeyGousset*	**HOL02** Full Day Hands-On Lab: Implementing Machine Learning Using C# and Visual Studio - *James McCaffrey*	**HOL03** Full Day Hands-On Lab: Modern Security Architecture for ASP.NET Core 3 - *Brock Allen*	**HOL04** Full Day Hands-On Lab: Hands-On with Cloud-Native .NET Development - *Rockford Lhotka*
Pre-Conference Workshops: Monday, March 2, 2020 *(Separate entry fee required)*					
7:30 AM	9:00 AM	Pre-Conference Hands-On Lab Registration - Coffee and Morning Pastries			
8:00 AM	5:00 PM	**M01** Workshop: Building Web Applications with ASP.NET Core and EF Core - *Philip Japikse*	**M02** Workshop: Design an App UX in a Day - *Billy Hollis*	**M03** Workshop: A Tour of Visual Studio 2019 - *Jason Bock*	
Day 1: Tuesday, March 3, 2020					
8:00 AM	9:15 AM	**T01** Angular Best Practices - Part 1 - *Deborah Kurata*	**T02** UX Design Fundamentals: What Do Your Users Really See? - *Billy Hollis*	**T03** Quantum Computing and the Future of Software Development - *Jerry Nixon*	**T04** Visual Studio Code for the Cloud Developer - *Chris Noring*
9:30 AM	10:45 AM	**T05** RxJS in Angular Best Practices - Part 2 - *Deborah Kurata*	**T06** Introduction to the Uno Platform - *Billy Hollis*	**T07** Introduction to ML.NET and AutoML - *James McCaffrey*	**T08** To Be Announced
11:00 AM	12:00 PM	**KEYNOTE: To Be Announced**			
12:00 PM	1:00 PM	Lunch - Platinum			
1:30 PM	2:45 PM	**T09** Securing APIs with ASP.NET Core 3, Visual Studio 2019, and IdentityServer4 - *Brock Allen*	**T10** Building Data Pipelines for Modern Data Warehouse with Spark and .NET in Azure - *Michael Rys*	**T11** Exploring Serverless Architecture - *Chris Noring*	**T12** Creating Business Applications Using Blazor (Razor Components) - *Michael Washington*
3:00 PM	4:15 PM	**T13** An Introduction to OpenID Connect and OAuth 2.0 using Azure B2C - *Brock Allen*	**T14** Building a Modern Data Warehouse in Microsoft Azure - *Michael Rys*	**T15** Building Business Applications Using Bots - *Michael Washington*	**T16** Build a DevOps Culture: Microsoft's Journey to Adopt an Agile Mindset and DevOps Culture - *Mickey Gousse*
4:15 PM	5:30 PM	Welcome Reception			
Day 2: Wednesday, March 4, 2020					
8:00 AM	9:15 AM	**W01** Getting Started with ASP.NET Core 3.X - *Philip Japikse*	**W02** Deep Dive: Blazing the Web — Building Web Applications in C# with Blazor - *Jason Bock*	**W03** *Storage Strategies in Microsoft Azure - Eric D. Boyd*	**W04** DevOps for Desktop Applications - *Oren Novotny*
9:30 AM	10:45 AM	**W05** JavaScript for the C# (and Java) Developer - *Philip Japikse*		**W06** Busy Developer's Guide to the Clouds - *Ted Neward*	**W07** Private and Public: Build & Release Custom NuGet Packages and more via Azure DevOps and GitHub - *Oren Novotny*
11:00 AM	12:00 PM	**GENERAL SESSION:** To Be Announced			
12:00 PM	1:00 PM	Birds-of-a-Feather Lunch			
1:30 PM	1:50 PM	**W08** Fast Focus: Getting Started with ASP.NET Core 2.0 Razor Pages - *Walt Ritscher*	**W09** Fast Focus: Migrate Your C# Code from .NET Framework to .NET Standard - *Rockford Lhotka*	**W10** Fast Focus: Onboard New Developers Faster with Custom Snippets! - *Jennifer Wadella*	
2:00 PM	2:20 PM	**W11** Fast Focus: Create Interactive Code Documentation with Try.NET - *Walt Ritscher*	**W12** Fast Focus: Serverless of Azure 101 - *Eric D. Boyd*	**W13** Fast Focus—Azure Pipelines Tips and Tricks - *Mickey Gousset*	
2:30 PM	3:45 PM	**W14** Supporting an Old- Fashioned AngularJS App in a Modern TypeScript World - *Jennifer Wadella*	**W15** Complex Event Processing and Messaging Patterns - *Eric D. Boyd*	**W16** Run Your Business Logic Everywhere with WebAssembly, Containers, and CSLA .NET - *Rockford Lhotka*	**W17** Deep Dive: Azure DevOps vs. GitHub—Fight! - *Brian Randell*
4:00 PM	5:15 PM	**W18** Busy Developer's Guide to Flutter - *Ted Neward*	**W19** An Introduction to Service Meshes on Kubernetes - *Marcel de Vries*	**W20** Moving Desktop Apps to .NET Core 3.0 - *Tim Corey*	
After Hours Fun		**Choose Your Las Vegas Adventure at VSLive!** - *High Roller Observation Wheel, FlyLINQ Zipline, or Eiffel Tower Viewing Deck*			
Day 3: Thursday, March 5, 2020					
8:00 AM	9:15 AM	**TH01** Human Readable JavaScript: Using ES2015+ to Craft Better Code - *Laurie Barth*	**TH02** Microsoft Power Platform, RAD Done Right: Building Automated Workflows with Microsoft - *Walt Ritscher*	**TH03** Modern Desktop Development with .NET Core 3.0 - *Tim Corey*	**TH04** Feature Flags for Better DevOps - *Mickey Gousset*
9:30 AM	10:45 AM	**TH05** The Multithreaded Web: A Tale of Workers - *Ramón Guijarro*	**TH06** The Definitive Guide to Organizing Community Events - *Jennifer Wadella*	**TH07** What's New in EF Core 3.0 - *Jim Wooley*	**TH08** A Software Engineer's Guide to DevOps - *Laurie Barth*
11:00 AM	12:15 PM	**TH09** Learn to Love Lamdas (and LINQ, Too!) - *Jeremy Clark*	**TH10** Get Your Poker Face On: How to Use Planning Poker to Slay Project Estimations - *Laura B. Janusek*	**TH11** Entity Framework Core Performance Monitoring and Tuning - *Jim Wooley*	**TH12** When "We Are Down" Is Not Good Enough. Site Reliability Engineering (SRE) in Azure - *René van Osnabrugge*
12:15 PM	1:15 PM	Lunch - *Platinium*			
1:15 PM	2:30 PM	**TH13** Run Faster: Parallel Programming in C# - *Jeremy Clark*	**TH14** Offices are Over: How & Why to Work Remotely in the Digital Age - *Laura B. Janusek*	**TH15** Deep Dive on Cosmos DB - *Leonard Lobel* [Combination of Azure Cosmos DB Part I—Introduction to Cosmos DB and Azure Cosmos DB Part II—Building Cosmos DB Applications]	**TH16** - Implementing Zero Downtime Application Deployments on Azure PaaS - *Marcel de Vries*
2:45 PM	4:00 PM	**TH17** Redux is Dead, Long Live Redux! - *Ramón Guijarro*	**TH18** How to Talk Like an Engineer - *Laurie Barth*		**TH19** Security in Your Pipelines. The Shift to Rugged DevOps - *René van Osnabrugge*
Post-Conference Workshops: Friday, March 6, 2020 *(Separate entry fee required)*					
8:00 AM	5:00 PM	**F01** Workshop: Building, Running & Continuously Deploying Microservices with Docker Containers on Azure - *Marcel de Vries & René van Osnabrugge*		**W02** Workshop: Developer Dive into SQL Server - *Leonard Lobel*	

Speakers and sessions subject to change

Virtual Track Key: Microsoft Sessions Foundational Sessions

Visit our website for Las Vegas updates on content and speakers! **vslive.com/lasvegas**

JAMES MCCAFFREY

Mixture Model Clustering Using C#

Data clustering is the process of grouping data items so that similar items are in the same group/cluster and dissimilar items are in different clusters. The most commonly used clustering algorithm is called k-means. The k-means approach is simple and effective, but it doesn't always work well with a dataset that has skewed distributions.

```
C:\MixtureModel\bin\Debug\MixtureModel.exe

Begin mixture model with C# demo

Data (height, width) to cluster looks like:
[0] 0.2000  0.7000
[1] 0.1000  0.9000
. . .
[7] 0.7000  0.1000

Setting K = 3, initializing w, a, u, V, Nk
Performing 5 E-M update iterations
Clustering done.

w (membership wts):
0.9207  0.0793  0.0000
0.9737  0.0263  0.0000
0.9587  0.0413  0.0000
0.0015  0.9962  0.0023
0.0000  0.9430  0.0570
0.0000  0.3848  0.6152
0.0000  0.1806  0.8194
0.0000  0.2750  0.7250

Nk (w column sums):
2.8545  2.9267  2.2188

a (mixture wts):
0.3568  0.3658  0.2774

u (cluster means):
0.1660  0.8017
0.5399  0.3986
0.7869  0.2005

V (cluster variances):
0.0023  0.0067
0.0373  0.0226
0.0083  0.0072

End mixture model demo
```

Figure 1 **Mixture Model Clustering Demo Run**

Code download available at msdn.com/magazine/1119magcode.

In this article I explain how to implement mixture model data clustering using the C# language. The best way to understand what mixture model clustering is and to see where this article is headed is to examine the demo program in **Figure 1**. The demo sets up a tiny dummy dataset with eight items. Each data item represents the height and width of a package of some sort. The first item is (0.2000, 0.7000) and the last item is (0.7000, 0.1000).

There are many variations of mixture model clustering. The variation presented in this article works only with numeric data and requires that the data be normalized or scaled so that all values are roughly in the same range. Normalization prevents large values, such as annual incomes, from overwhelming small values such as high school GPAs.

The demo program specifies the number of clusters as K = 3. Most clustering techniques, including mixture model clustering, require you to set the number of clusters, and this must be determined by trial and error. Mixture model clustering uses a technique called expectation-maximization (EM) optimization, which is an iterative process. The demo iterates five times.

The final clustering result is contained in a matrix called membership weights, w. The w matrix has eight rows. Each row of w corresponds to a row of the dataset.

The first row of w is (0.9207, 0.0793, 0.0000). Notice that the values in each row of w sum to 1.0 so they can be loosely interpreted as the probability that the corresponding data item belongs to a cluster. Therefore, data item [0] belongs to cluster 0 because column [0] of w has the largest value. Similarly, the value of w[7] is (0.0000, 0.2750, 0.7250), so data item [7] belongs to cluster 2.

The demo program displays the values of internal data structures named Nk (column sums of w), a (mixture weights), u (cluster means) and V (cluster variances). These values are used by the EM optimization algorithm part of mixture model clustering.

This article assumes you have intermediate or better programming skills with C#, but doesn't assume you know anything about mixture model data clustering. The demo is coded using C#, but you shouldn't have any trouble refactoring the code to another language such as JavaScript or Visual Basic. The complete demo code is presented in this article. The source code is also available in the accompanying download.

Understanding the Multivariate Gaussian Distribution

Mixture model clustering and EM are really general techniques rather than specific algorithms. The key math equations for the

variation of mixture model clustering and EM optimization used in this article are shown in **Figure 2**. Depending on your math background, the equations may appear intimidating, but they're actually quite simple, as you'll see shortly.

The key to the variation of mixture model clustering presented here is understanding the multivariate Gaussian (also called normal) probability distribution. And to understand the multivariate Gaussian distribution, you must understand the univariate Gaussian distribution.

Suppose you have a dataset that consists of the heights (in inches) of men who work at a large tech company. The data would look like 70.5, 67.8, 71.4 and so forth (all inches). If you made a bar graph for the frequency of the heights, you'd likely get a set of bars where the highest bar is for heights between 68.0 to 72.0 inches (frequency about 0.6500). You'd get slightly lower bars for heights between 64.0 to 68.0 inches, and for 72.0 inches to 76.0 inches (about 0.1400 each), and so on. The total area of the bars would be 1.0. Now if you drew a smooth line connecting the tops of the bars, you'd see a bell-shaped curve.

The univariate Gaussian distribution is a mathematical abstraction of this idea. The exact shape of the bell-shaped curve is determined by the mean (average) and variance (measure of spread). The math equation that defines the bell-shaped curve of a univariate Gaussian distribution is called the probability density function (PDF). The math equation for the PDF of a univariate Gaussian distribution is shown as equation (1) in **Figure 2**. The variance is given by sigma squared. For example, if x = 1.5, u = 0, and var = 1.0 then f(x, u, var) = 0.1295, which is the height of the bell-shaped curve at point x = 1.5.

The value of the PDF for a given value of x isn't a probability. The probability of a range of values for x is the area under the curve of the PDF function, which can be determined using calculus. Even though a PDF value by itself isn't a probability, you can compare PDF values to get relative likelihoods. For example, if PDF(x1) = 0.5000 and PDF(x2) = 0.4000 you can conclude that the probability of x1 is greater than the probability of x2.

Now, suppose your dataset has items where each has two or more values. For example, instead of just the heights of men, you have (height, weight, age). The math abstraction of this scenario is called the multivariate Gaussian distribution. The dimension, d, of a multivariate Gaussian distribution is the number of values in each data item, so (height, weight, age) data has dimension d = 3.

The equation for the PDF of a multivariate Gaussian distribution is shown as equation (2) in **Figure 2**. The x is in bold to indicate it's a multivalued vector. For a multivariate Gaussian distribution, instead of a variance, there is a (d x d) shape covariance matrix, represented by Greek letter uppercase sigma. Calculating the PDF for a univariate x is simple, but calculating the PDF for a multivariate x is extremely difficult because you must calculate the determinant and inverse of a covariance matrix.

Instead of using the multivariate Gaussian distribution, you can greatly simplify mixture model calculations by assuming that each component of a multivalued data item follows an independent univariate Gaussian distribution. For example, suppose d = 3 and x = (0.4, 0.6, 0.8). Instead of calculating a multivalued mean and a (d x d) multivalued covariance matrix and then using equation (2), you can look at each data component separately. This would give you three means, each a single value, and three variances, each also a single value. Then you can calculate three separate PDF values using equation (1) and average the three PDF values to get a final PDF value.

$$(1)\quad f(x,\mu,\sigma^2) = \frac{1}{\sqrt{2\pi * \sigma^2}} * e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$(2)\quad f(\boldsymbol{x},\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^d * det(\boldsymbol{\Sigma})}} * e^{-\frac{1}{2}*[(\boldsymbol{x}-\boldsymbol{\mu})^T \cdot inv(\boldsymbol{\Sigma}) \cdot (\boldsymbol{x}-\boldsymbol{\mu})]}$$

$$(3)\quad w_{ik} = \frac{f(x_i,\mu_k,\sigma_k^2) * a_k}{\sum_{m=1}^{K} f(x_i,\mu_k,\sigma_k^2) * a_m}$$

$$(4)\quad \alpha_k = \frac{N_k}{N}, \qquad N_k = \sum_{i=1}^{N} w_{ik}$$

$$(5)\quad \mu_k = \frac{1}{N_k}\sum_{i=1}^{N} w_{ik} * x_i$$

$$(6)\quad \sigma_k^2 = \frac{1}{N_k}\sum_{i=1}^{N} w_{ik} * (x_i - \mu_k)^2$$

Figure 2 **Mixture Model Expectation-Maximization Optimization Equations**

This technique is called a Gaussian mixture model because it assumes the data is clustered according to a mixture of K multivariate Gaussian probability distributions. It's known as a naive approach because the assumption that the data components are mathematically independent of each other is rarely true for real-life data. However, the naive Gaussian PDF approach is dramatically simpler than calculating a true multivariate Gaussian PDF, and the naive technique often (but not always) works well in practice.

The Demo Program

The complete demo program, with several minor edits to save space, is presented in **Figure 3**. To create the program, I launched Visual Studio and created a new console application named MixtureModel. I used Visual Studio 2017, but the demo has no significant .NET Framework dependencies.

After the template code loaded, at the top of the editor window I removed all unneeded namespace references, leaving just the reference to the top-level System namespace. In the Solution Explorer window, I right-clicked on file Program.cs, renamed it to the more descriptive MixtureModelProgram.cs and allowed Visual Studio to automatically rename class Program.

All normal error checking has been omitted to keep the main ideas as clear as possible. I used static methods and class-scope constants rather than OOP for simplicity.

Visual Studio LIVE! EXPERT SOLUTIONS FOR ENTERPRISE DEVELOPERS | SQL Server LIVE! TRAINING FOR DBAs AND IT PROS | TECHMENTOR IN-DEPTH TRAINING FOR IT PROS | Office & SharePoint LIVE! ON-PREMISE, CLOUD, & CROSS-PLATFORM TRAINING | Artificial Intelligence LIVE! AI FOR DEVELOPERS AND DATA SCIENTISTS | Cloud & Containers LIVE! CLOUD-NATIVE, PAAS & SERVERLESS COMPUTING

The Ultimate Education Destination

LIVE! 360
TECH EVENTS WITH PERSPECTIVE

6 Great Events, 1 Low Price!

November 17-22, 2019 | ORLANDO
Royal Pacific Resort at Universal

CHOOSE LIVE! 360 FOR:

- 202 Technical Sessions
- 8 Keynote Presentations
- 6 Panel Discussions
- 21 In-depth Workshops
- 8 Hands-on Labs
- 120 Speakers
- ∞ Networking Opportunities

live360events.com/orlando

Live! 360 brings the IT and developer community together for a unique conference, featuring 6 co-located conferences for (almost) every IT title. Customize your learning by choosing from hundreds of sessions, dozens of track topics, workshops and hands-on labs from hundreds of expert speakers.

EVENT PARTNER: Microsoft
PLATINUM SPONSOR: accusoft
SILVER SPONSOR: aws
SUPPORTED BY: msdn magazine | Redmond Channel Partner | Redmond Magazine | Virtualization & Cloud Review | Visual Studio Magazine

Visual Studio LIVE!

EXPERT SOLUTIONS FOR ENTERPRISE DEVELOPERS

Visual Studio Live! features unbiased and practical developer training on the Microsoft Platform. Explore hot topics such as ASP.NET Core, JavaScript, Xamarin, Database Analytics and more!

SQL Server LIVE!

TRAINING FOR DBAs AND IT PROS

Whether you are a DBA, developer, IT Pro or Analyst, SQL Server Live! will provide you with the skills you need to drive your data to succeed.

Office & SharePoint LIVE!

ON-PREMISE, CLOUD, & CROSS-PLATFORM TRAINING

Office & SharePoint Live! provides leading-edge training to administrators and developers who must customize, deploy and maintain SharePoint Server on-premises and in Office 365.

TECHMENTOR

IN-DEPTH TRAINING FOR IT PROS

TechMentor gives a strong emphasis on doing more with the tech you already own plus solid coverage of what is next - striking the perfect balance of training essentials for the everyday IT Pro.

Artificial Intelligence LIVE!

AI FOR DEVELOPERS AND DATA SCIENTISTS

Artificial Intelligence Live! offers real-world training on the languages, libraries, APIs, tools and cloud services you need to implement real AI and Machine Learning solutions today and into the future.

NEW! IN 2019 Cloud & Containers LIVE!

CLOUD-NATIVE, PAAS & SERVERLESS COMPUTING

Cloud & Containers Live! covers both IT infrastructure and software development aspects of cloud-native software design, development, release, deployment, operations, instrumentation/monitoring and maintenance.

Who Should Attend:

- Developers
- IT Professionals
- Database Administrators
- Business Intelligence Professionals
- SharePoint Professionals
- and more!

Seats Are Filling Quickly – Secure Your Spot Today!!

Promo Code MSDN

live360events.com/orlando

PRODUCED BY CONVERGE360 an 1105 MEDIA company

CONNECT WITH LIVE! 360

twitter.com/live360
@live360

facebook.com
Search "Live 360"

instagram.com
@live360_events

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

Figure 3 **The Mixture Model Clustering Demo Program**

```
using System;
namespace MixtureModel
{
  class MixtureModelProgram
  {
    const int N = 8; const int K = 3; const int d = 2;

    static void Main(string[] args)
    {
      Console.WriteLine("Begin mixture model with demo ");
      double[][] x = new double[N][];  // 8 x 2 data
      x[0] = new double[] { 0.2, 0.7 };
      x[1] = new double[] { 0.1, 0.9 };
      x[2] = new double[] { 0.2, 0.8 };
      x[3] = new double[] { 0.4, 0.5 };
      x[4] = new double[] { 0.5, 0.4 };
      x[5] = new double[] { 0.9, 0.3 };
      x[6] = new double[] { 0.8, 0.2 };
      x[7] = new double[] { 0.7, 0.1 };

      Console.WriteLine("Data (height, width): ");
      Console.Write("[0] "); VectorShow(x[0]);
      Console.Write("[1] "); VectorShow(x[1]);
      Console.WriteLine(". . . ");
      Console.Write("[7] "); VectorShow(x[7]);

      Console.WriteLine("K=3, initing w, a, u, S, Nk");
      double[][] w = MatrixCreate(N, K);
      double[] a = new double[K] { 1.0/K, 1.0/K, 1.0/K };
      double[][] u = MatrixCreate(K, d);
      double[][] V = MatrixCreate(K, d, 0.01);
      double[] Nk = new double[K];

      u[0][0] = 0.2; u[0][1] = 0.7;
      u[1][0] = 0.5; u[1][1] = 0.5;
      u[2][0] = 0.8; u[2][1] = 0.2;

      Console.WriteLine("Performing 5 E-M iterations ");
      for (int iter = 0; iter < 5; ++iter) {
        UpdateMembershipWts(w, x, u, V, a);  // E step
        UpdateNk(Nk, w);  // M steps
        UpdateMixtureWts(a, Nk);
        UpdateMeans(u, w, x, Nk);
        UpdateVariances(V, u, w, x, Nk);
      }
      Console.WriteLine("Clustering done. \n");
      ShowDataStructures(w, Nk, a, u, V);

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

    static void ShowDataStructures(double[][] w,
      double[] Nk, double[] a, double[][] u, double[][] V)
    {
      Console.WriteLine("w:"); MatrixShow(w, true);
      Console.WriteLine("Nk:"); VectorShow(Nk, true);
      Console.WriteLine("a:"); VectorShow(a, true);
      Console.WriteLine("u:"); MatrixShow(u, true);
      Console.WriteLine("V:"); MatrixShow(V, true);
    }

    static void UpdateMembershipWts(double[][] w,
      double[][] x, double[][] u, double[][] V, double[] a)
    {
      for (int i = 0; i < N; ++i) {
        double rowSum = 0.0;
        for (int k = 0; k < K; ++k) {
          double pdf = NaiveProb(x[i], u[k], V[k]);
          w[i][k] = a[k] * pdf;
          rowSum += w[i][k];
        }
        for (int k = 0; k < K; ++k)
          w[i][k] = w[i][k] / rowSum;
      }
    }

    static void UpdateNk(double[] Nk, double[][] w)
    {
      for (int k = 0; k < K; ++k) {
        double sum = 0.0;
        for (int i = 0; i < N; ++i)
          sum += w[i][k];
        Nk[k] = sum;
      }
    }

    static void UpdateMixtureWts(double[] a, double[] Nk)
    {
      for (int k = 0; k < K; ++k)
        a[k] = Nk[k] / N;
```

The Data Structures

The demo code sets constants N = 8 (number of data items), d = 2 (data dimensionality), and K = 3 (number of clusters). There are six data structures. Matrix X is an array-of-arrays with size (8 x 2) and it holds the data. In a non-demo scenario you'd load normalized data into memory from a text file.

Matrix w, which has size (8 x 3), holds the membership weights and the clustering information. Each row of w represents a data item, and each column of w corresponds to a cluster. Mixture model clustering assumes that each cluster follows some probability distribution. The most commonly assumed distribution is the multivariate Gaussian, so the technique is called Gaussian mixture model (GMM). The demo uses a simplified Gaussian, so I call the technique naive Gaussian mixture model, but this isn't a standard name.

Helper array Nk has 3 cells and holds the sums of the 3 columns of w. Array a has 3 cells and is called the mixture weights. The values in array a sum to 1.0 and are weights that indicate the contribution of each Gaussian distribution to the model. For example, if a = (0.30, 0.60, 0.10), then the first Gaussian distribution contributes 30 percent to the mixture. Each cell of the a array is initialized to 1 / 3 = 0.3333 so that each Gaussian initially contributes equally.

Matrices u and V have size (3 x 2) and hold the univariate means and variances. For example, if u[1][0] = 0.55, the mean for cluster 1, data component 0 (package height), is 0.55. If V [2][1] = 0.33, the variance for cluster 2, data component 1 (package width), is 0.33.

Updating Membership Weights

The membership weights stored in matrix w are updated in each EM iteration according to equation (3) in **Figure 2**. In words, for each N = 8 rows of w, you compute the K = 3 naive probabilities using the corresponding values of X, u, and V and then multiply each naive probability by the corresponding mixture weight stored in the a vector. After you compute the weighted probabilities for a row, each of the values in the row is normalized by dividing by the row sum.

Updating the membership weights is the expectation step (often shortened to E-step) of the EM optimization algorithm. After membership weights in w are updated, the Nk column sums and mixture weights in vector a are updated using equation (4), the means in matrix u are updated using equation (5), and the variances in matrix V are updated using equation (6) in **Figure 2**. Updating Nk, a, u and V is the maximization step of EM optimization. The idea is to find the values in a, u and V that best match the data.

```
  }

  static void UpdateMeans(double[][] u, double[][] w,
    double[][] x, double[] Nk)
  {
    double[][] result = MatrixCreate(K, d);
    for (int k = 0; k < K; ++k) {
      for (int i = 0; i < N; ++i)
        for (int j = 0; j < d; ++j)
          result[k][j] += w[i][k] * x[i][j];

      for (int j = 0; j < d; ++j)
        result[k][j] = result[k][j] / Nk[k];
    }

    for (int k = 0; k < K; ++k)
      for (int j = 0; j < d; ++j)
        u[k][j] = result[k][j];
  }

  static void UpdateVariances(double[][] V, double[][] u,
    double[][] w, double[][] x, double[] Nk)
  {
    double[][] result = MatrixCreate(K, d);
    for (int k = 0; k < K; ++k) {
      for (int i = 0; i < N; ++i)
        for (int j = 0; j < d; ++j)
          result[k][j] += w[i][k] * (x[i][j] - u[k][j]) *
            (x[i][j] - u[k][j]);

      for (int j = 0; j < d; ++j)
        result[k][j] = result[k][j] / Nk[k];
    }

    for (int k = 0; k < K; ++k)
      for (int j = 0; j < d; ++j)
        V[k][j] = result[k][j];
  }

  static double ProbDenFunc(double x, double u, double v)
  {
    // Univariate Gaussian
    if (v == 0.0) throw new Exception("0 in ProbDenFun");
    double left = 1.0 / Math.Sqrt(2.0 * Math.PI * v);
    double pwr = -1 * ((x - u) * (x - u)) / (2 * v);
    return left * Math.Exp(pwr);
  }

  static double NaiveProb(double[] x, double[] u,
    double[] v)
  {
    // Poor man's multivariate Gaussian PDF
    double sum = 0.0;
    for (int j = 0; j < d; ++j)
      sum += ProbDenFunc(x[j], u[j], v[j]);
    return sum / d;
  }

  static double[][] MatrixCreate(int rows, int cols,
    double v = 0.0)
  {
    double[][] result = new double[rows][];
    for (int i = 0; i < rows; ++i)
      result[i] = new double[cols];
    for (int i = 0; i < rows; ++i)
      for (int j = 0; j < cols; ++j)
        result[i][j] = v;
    return result;
  }

  static void MatrixShow(double[][] m, bool nl = false)
  {
    for (int i = 0; i < m.Length; ++i) {
      for (int j = 0; j < m[0].Length; ++j) {
        Console.Write(m[i][j].ToString("F4") + "  ");
      }
      Console.WriteLine("");
    }
    if (nl == true)
      Console.WriteLine("");
  }

  static void VectorShow(double[] v, bool nl = false)
  {
    for (int i = 0; i < v.Length; ++i)
      Console.Write(v[i].ToString("F4") + "  ");
    Console.WriteLine("");
    if (nl == true)
      Console.WriteLine("");
  }
 } // Program class
} // ns
```

Wrapping Up

The code and most of the math notation used in this article are based on a short paper titled "The EM Algorithm for Gaussian Mixtures." The paper isn't on a stable Web site and has no author given, but the subtitle and URL indicate the paper is part of lecture notes from a UC Irvine computer science class. You can easily find the paper as a .pdf file by doing an Internet search for the title. I used this somewhat obscure paper because it's clear and accurate, and almost all of the other Gaussian mixture model resources I found had significant technical errors.

Data clustering is usually an exploratory process. Clustering results are typically examined visually to see if any interesting patterns appear. Most of my colleagues start a clustering investigation by looking at a graph of the source data. If the data appears evenly distributed, then using the k-means algorithm (or one of its many variations) usually works well. But if the data appears skewed, a mixture model clustering approach such as the one presented in this article often gives better results. Most clustering algorithms, including naive Gaussian mixture model, are extremely sensitive to initial values of the means and variances, so these values are often supplied via a preliminary k-means analysis. ■

Aloha, Servus and Ciao

It's been a tremendous pleasure writing articles about software development, testing and machine learning for *MSDN Magazine* over the past 17 years. Rather than look back, I prefer to look forward. The essence of *MSDN Magazine* has always been filling the gap between low-level software documentation (too much detail) and high-level Hello World-style examples (not enough detail). The need to fill that gap won't go away so I speculate that the soul of *MSDN Magazine* will quickly reemerge in an online format of some sort. So goodbye/hello for now and keep your eyes open for *MSDN Magazine* authors and editors providing useful information in a new format to the software developer community.

– James McCaffrey

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

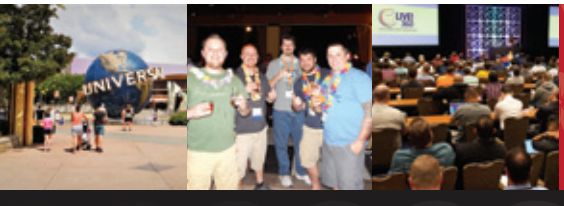
Thanks to the following Microsoft technical experts who reviewed this article: *Ricky Loynd, Kirk Olynyk*

Data. Driven.

At SQL Server Live!, we want to help DBAs, analytics experts, systems administrators, and developers like you do more with your SQL Server and Microsoft Data Platform investment. We want to help you improve performance, get insights from your data, step up security, and take advantage of new features across the platform. We want you to master reporting, BI, data integration, and developer tools and techniques. We will show attendees how to adopt new techniques, improve old approaches, explore and visualize data, and modernize SQL Server infrastructure. We teach and demonstrate how to integrate cloud-based data services, run SQL Server technology in the cloud, use Power BI and Analysis Services, and plan for SQL Server recovery and availability.

TRACK TOPICS INCLUDE:

- **Business Intelligence**
- **SQL Server Administration & Maintenance**
- **SQL Server for Developers**
- **SQL Server Features & Components**
- **SQL Server Performance Tuning & Optimization**



Seats Are Filling Up Quickly – Secure Your Spot Today!

Use Promo Code MSDN

A Part of Live! 360: The Ultimate Education Destination

6 Great Events, 1 Low Price!

Visual Studio LIVE! — EXPERT SOLUTIONS FOR ENTERPRISE DEVELOPERS
SQL Server LIVE! — TRAINING FOR DBAs AND IT PROS
TECHMENTOR — IN-DEPTH TRAINING FOR IT PROS
Office & SharePoint LIVE! — ON-PREMISE, CLOUD, & CROSS-PLATFORM TRAINING
Artificial Intelligence LIVE! — AI FOR DEVELOPERS AND DATA SCIENTISTS
Cloud & Containers LIVE! — CLOUD-NATIVE, PAAS & SERVERLESS COMPUTING

SQLLive360.com

EVENT PARTNER: Microsoft
PLATINUM SPONSOR: accusoft
SILVER SPONSOR: aws
SUPPORTED BY: msdn magazine, Redmond Channel Partner, Redmond Magazine, Virtualization & Cloud Review, Visual Studio Magazine

AGENDA-AT-A-GLANCE

#LIVE360

Business Intelligence	SQL Server Administration & Maintenance	SQL Server Features & Components	SQL Server for Developers	SQL Server Performance Tuning and Optimization

START TIME	END TIME			
SQL Server Live! Full Day Hands-On Lab: Sunday, November 17, 2019				
7:15 AM	9:00 AM	Registration • Coffee and Morning Pastries		
9:00 AM	6:00 PM	**SQS01** Hands-On Lab: The A to Z of Cloud-based SQL Server Solutions - *Allan Hirt*		
2:00 PM	7:00 PM	Pre-Conference Registration - Royal Pacific Resort Conference Center		
START TIME	**END TIME**	**SQL Server Live! Pre-Conference Workshops: Monday, November 18, 2019**		
7:00 AM	8:30 AM	Registration • Coffee and Morning Pastries		
8:30 AM	5:30 PM	**SQM01** Workshop: Advanced Data Protection - Security and Privacy in SQL Server - *Thomas LaRock & Karen Lopez*	**SQM02** Workshop: SQL Server 2019 - *Leonard Lobel*	
6:30 PM	8:00 PM	Dine-A-Round Dinner @ Universal CityWalk - 6:30pm - Meet at Conference Registration Desk to walk over with the group		
START TIME	**END TIME**	**SQL Server Live! Day 1: Tuesday, November 19, 2019**		
7:00 AM	8:00 AM	Registration • Coffee and Morning Pastries		
8:00 AM	9:00 AM	**SQL SERVER LIVE! KEYNOTE: SQL Server 2019 Big Data Clusters – Unifying Relational and Big Data** *Rony Chatterjee, Senior Program Manager on Azure Data, Microsoft*		
9:15 AM	10:30 AM	**SQT01** AWS Versus Azure: Data Services Comparison - *Thomas LaRock*	**SQT02** Availability Fundamentals for SQL Server - *Allan Hirt*	**SQT03** Power BI: What Have You Done For Me Lately - *Andrew Brust*
10:30 AM	11:00 AM	Networking Break • Visit the EXPO - *Pacifica 7*		
11:00 AM	12:00 PM	**LIVE! 360 KEYNOTE: The Power of Real World DevOps** - *Jessica Deen, Azure Avenger, Microsoft & Abel Wang, Senior Cloud Developer Advocate, Microsoft*		
12:00 PM	12:45 PM	Lunch • Visit the EXPO		
12:45 PM	1:30 PM	Dessert Break • Visit the EXPO		
1:30 PM	1:50 PM	**SQT04** Fast Focus: Automation for the DBA: Embrace Your Inner Sloth - *William Durkin*	**SQT05** Fast Focus: Who's Tinkling in Your Data Lake? - *Karen Lopez*	
2:00 PM	2:20 PM	**SQT06** Fast Focus: From Adaptive to Intelligent: Query Processing in SQL 2019 - *Hugo Kornelis*	**SQT07** Fast Focus: Performance Tuning Without Changing Code - *Thomas LaRock*	
2:20 PM	2:45 PM	Networking Break • Visit the EXPO - *Pacifica 7*		
2:45 PM	4:00 PM	**SQT08** SQL Server 2019 Deep Dive - *Scott Klein*	**SQT09** SQL Server In-Memory Database Objects - *Denny Cherry*	**SQT10** What's New in SQL Server 2019 for Analytics - *Thomas LeBlanc*
4:15 PM	5:30 PM	**SQT11** Microsoft's New Database Experimentation Assistant (DEA) - *Mindy Curnutt*	**SQT12** Common Troubleshooting Techniques for Availability Groups and Failover Cluster Instances - *Allan Hirt*	**SQT13** Power BI implementation guidance: from Pro to Premium and Desktop to Embedded - *Thomas LeBlan*
5:30 PM	7:30 PM	Exhibitor Reception - *Pacifica 7*		
START TIME	**END TIME**	**SQL Server Live! Day 2: Wednesday, November 20, 2019**		
7:30 AM	8:00 AM	Registration • Coffee and Morning Pastries		
8:00 AM	9:15 AM	**SQW01** An Introduction to Spatial Data in SQL Server - *Mindy Curnutt*	**SQW02** It's Broken, Now What?! Practical Problem Solving - *William Durkin*	**SQW03** Getting Started with Apache Spark - *Kevin Feasel*
9:30 AM	10:45 AM	**SQW04** Blockchain for the DBA & Data Professional - *Karen Lopez*	**SQW05** SQL Server Open Query Store - *William Durkin*	**SQW06** PolyBase in Action - *Kevin Feasel*
10:45 AM	11:30 AM	Networking Break • Visit the EXPO - *Pacifica 7*		
11:30 AM	12:30 PM	**LIVE! 360 KEYNOTE: Why So Serious? The Importance of Comedy in Tech** - *Chloe Condon, Senior Cloud Advocate, Microsoft*		
12:30 PM	1:30 PM	Birds-of-a-Feather Lunch		
1:30 PM	2:00 PM	Dessert Break • Visit the EXPO		
2:00 PM	3:15 PM	**SQW07** SQL Data Discovery and Classification - *Karen Lopez*	**SQW08** Things You Should Never Do In Microsoft SQL Server - *Denny Cherry*	**SQW09** Deep Dive into Power BI Administration and Security Internals - *Ginger Grant*
3:15 PM	4:00 PM	Networking Break • Visit the EXPO • Expo Raffle @ 3:30 p.m. - *Pacifica 7*		
4:00 PM	5:15 PM	**SQW10** Migrating SSIS Workloads to Azure Data Factory - *Joshua Luedeman*	**SQW11** Improve Your Database Performance in Seven Simple Steps - *Hugo Kornelis*	**SQW12** Performance Tuning Power BI - Ginger Grant
7:30 PM	9:00 PM	Live! 360 Dessert Luau - *Wantilan Pavilion*		
START TIME	**END TIME**	**SQL Server Live! Day 3: Thursday, November 21, 2019**		
7:30 AM	8:00 AM	Registration • Coffee and Morning Pastries		
8:00 AM	9:15 AM	**SQH01** Deep Dive into Adaptive Query Processing - *Hugo Kornelis*	**SQH02** Using Modular Scripts to Perform SQL Compliance Audits in Seconds - *Chris Bell*	**SQH03** DataOps and Business Intelligence Testing Approaches - *Jen Stirrup*
9:30 AM	10:45 AM	**SQH04** Data Security and Privacy Techniques for Modern Databases - *Thomas LaRock*	**SQH05** Building Your First SQL Server Container Lab in Docker - *Chris Bell*	**SQH06** To Be Announced
11:00 AM	12:00 PM	**SQL SERVER LIVE! PANEL DISCUSSION: SQL Server is Dead, Long Live SQL Server!** *Thomas LaRock (moderator), Karen Lopez, Bradley Ball, Jen Stirrup, Joseph D'Antoni*		
12:00 PM	1:00 PM	Lunch - *Oceana Ballroom*		
1:00 PM	2:15 PM	**SQH07** Machine Learning with R in Azure SQL Database - *Bradley Ball*	**SQH08** Building a Better Data Solution: Microsoft SQL Server and Azure Data Services - *Joseph D'Antoni*	**SQH09** Data Preparation: A framework with Power BI, Power Query, and Power BI Dataflows - *Jen Stirrup*
2:30 PM	3:45 PM	**SQH10** TimescaleDB on Azure Database for PostgreSQL - *Bradley Ball*	**SQH11** Containers, Pods, and Databases - Learning About the Future of Infrastructure - *Joseph D'Antoni*	**SQH12** Getting Started with Power BI Report Server - *Joshua Luedeman*
4:00 PM	5:00 PM	Next? Live! 360 Networking Event *Thomas LaRock, Bradley Ball, Joseph D'Antoni, Karen Lopez & Jen Stirrup*		
START TIME	**END TIME**	**SQL Server Live! Post-Conference Workshops: Friday, November 22, 2019**		
7:30 AM	8:00 AM	Registration • Coffee and Morning Pastries		
8:00 AM	5:00 PM	**SQF01** Workshop: Azure SQL Data Warehouse: A Full Day of the Transformative New NDA Features - *Bradley Ball, Joshua Luedeman, & Gareth Swanepoel*	**SQF02** Workshop: SQL Server High Performance Development - *Joseph D'Antoni*	

Speakers and sessions subject to change

PRODUCED BY
CONVERGE360 an 1105 MEDIA company

CONNECT WITH LIVE! 360

twitter.com/live360
@live360

facebook.com
Search "Live 360"

instagram.com
@live360_events

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

TECHMENTOR

IN-DEPTH TRAINING FOR IT PROS

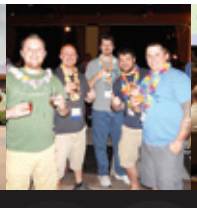
November 17-22, 2019 | ORLANDO
Royal Pacific Resort at Universal

Where In-Depth IT Training Meets the Sunshine

TechMentor offers in-depth training for IT Pros, giving you the perfect balance of the tools you need today, while preparing you for tomorrow. Expect troubleshooting tips, performance optimization training, and best practices from peers and experts in the industry. Plus, there will be dedicated coverage of Windows PowerShell, core Windows Server functionality, Security, System Center and so much more. We'll see you in the sun!

TRACK TOPICS INCLUDE:

- Client and Endpoint Management
- PowerShell and DevOps
- Infrastructure
- Soft Skills for ITPros
- Security and Ethical Hacking
- Azure (Public/Hybrid)
- Office 365 for the IT Pro



Seats Are Filling Up Quickly – Secure Your Spot Today!
Use Promo Code MSDN

A Part of Live! 360: The Ultimate Education Destination

6 Great Events, 1 Low Price!

Visual Studio LIVE! — EXPERT SOLUTIONS FOR ENTERPRISE DEVELOPERS
SQL Server LIVE! — TRAINING FOR DBAs AND IT PROS
TECHMENTOR — IN-DEPTH TRAINING FOR IT PROS
Office & SharePoint LIVE! — ON-PREMISE, CLOUD, & CROSS-PLATFORM TRAINING
Artificial Intelligence LIVE! — AI FOR DEVELOPERS AND DATA SCIENTISTS
Cloud & Containers LIVE! — CLOUD-NATIVE, PAAS & SERVERLESS COMPUTING

techmentorevents.com/orlando

EVENT PARTNER: Microsoft
PLATINUM SPONSOR: accusoft
SILVER SPONSOR: aws
SUPPORTED BY: msdn magazine, Redmond Channel Partner, Redmond Magazine, Virtualization & Cloud Review, Visual Studio Magazine

AGENDA-AT-A-GLANCE

#LIVE360

Client and Endpoint Management	PowerShell and DevOps	Infrastructure	Soft Skills for IT Pros	Security	Azure (Public/Hybrid)	Office 365 for the IT Pro

START TIME	END TIME	
TechMentor Full Day Hands-On Labs: Sunday, November 17, 2019		
7:15 AM	9:00 AM	Registration • Coffee and Morning Pastries
9:00 AM	6:00 PM	**TMS01** Hands-On Lab: Building a Bulletproof Privileged Access Workstation (PAW) - *Sami Laiho*
		TMS02 Hands-On Lab: Rock JEA (Just Enough Administration) the Easy Way with Windows Admin Center - *John O'Neill Sr.*
2:00 PM	7:00 PM	Pre-Conference Registration - Royal Pacific Resort Conference Center
TechMentor Pre-Conference Workshops: Monday, November 18, 2019		
7:00 AM	8:30 AM	Registration • Coffee and Morning Pastries
8:30 AM	5:30 PM	**TMM01** Workshop: Azure from IAAS to PAAS—A Demo-Filled Workshop - *Koenraad Haedens*
		TMM02 Workshop: PowerShell Remoting Deep Dive - *Jeffery Hicks*
6:30 PM	8:00 PM	Dine-A-Round Dinner @ Universal CityWalk - 6:30pm
TechMentor Day 1: Tuesday, November 19, 2019		
7:00 AM	8:00 AM	Registration • Coffee and Morning Pastries
8:00 AM	9:00 AM	**TECHMENTOR KEYNOTE: Getting Employees to (Really) Adopt New Technology** - *Stephen Rose, Marketing and Storytelling Lead, Microsoft 365 Team, Microsoft*
9:15 AM	10:30 AM	**TMT01** PowerShell Tools and Techniques Basics for IT Pros, Not Devs - *John O'Neill Sr.*
		TMT02 Network Sustainability and Cyber Security Defense - *Omar Valerio*
		TMT03 Zero Trust: Trust Identities, Not Your Network - *Ricky Pullan & Swetha Rai*
10:30 AM	11:00 AM	Networking Break • Visit the EXPO - *Pacifica 7*
11:00 AM	12:00 PM	**LIVE! 360 KEYNOTE: The Power of Real World DevOps** - *Jessica Deen, Azure Avenger, Microsoft & Abel Wang, Senior Cloud Developer Advocate, Microsoft*
12:00 PM	12:45 PM	Lunch • Visit the EXPO
12:45 PM	1:30 PM	Dessert Break • Visit the EXPO
1:30 PM	1:50 PM	**TMT04** Fast Focus: The 10 Handiest PowerShell Cmdlets for Managing Hyper-V - *John O'Neill Sr.*
		TMT05 Fast Focus: Rock Configuration Manager in 20 Minutes - *Émile Cabot*
2:00 PM	2:20 PM	**TMT06** Fast Focus: PowerShell Fast & Furious - *Jeffery Hicks*
		TMT07 Fast Focus: Be a Better Speaker—How to Create Presentations that Inspire and Make People Care - *Erwin Derksen*
2:20 PM	2:45 PM	Networking Break • Visit the EXPO - Pacifica 7
2:45 PM	4:00 PM	**TMT08** An Inclusive Look at Fighting Phishing - *Roger Grimes*
		TMT09 60 Life Hacks of the Windows OS in 75 Minutes - *Sami Laiho*
		TMT10 Azure Windows Virtual Desktop – Deploy and Configure in 50 Minutes - Koenraad Haedens
4:15 PM	5:30 PM	**TMT11** Fast-Track Your Way to IT Pro Rockstar Status - *John O'Neill Sr. & Cristal Kawula*
		TMT12 Deploying and Managing Office 365 ProPlus in a Modern World - *Peter Daalmans*
		TMT13 Secure Remote Management with Just Enough Administration - *Jeffery Hicks*
5:30 PM	7:30 PM	Exhibitor Reception - Pacifica 7
TechMentor Day 2: Wednesday, November 20, 2019		
7:30 AM	8:00 AM	Registration • Coffee and Morning Pastries
8:00 AM	9:15 AM	**TMW01** 10 Ways to Hack You Via Email - *Roger Grimes*
		TMW02 How YOU Deploy Office ProPlus / 2019 in the Enterprise Successfully - *Erwin Derksen*
		TMW03 Windows Powershell and Workflow: A Powerful Pairing! - *Michael Wiley*
9:30 AM	10:45 AM	**TMW04** Troubleshooting the Modern Managed Client - *Panu Saukko*
		TMW05 Managing Android and IOS in the Enterprise - *Peter Daalmans*
		TMW06 Become a PowerShell Debugging Ninja! - *Kirk Munro*
10:45 AM	11:30 AM	Networking Break • Visit the EXPO - *Pacifica 7*
11:30 AM	12:30 PM	**LIVE! 360 KEYNOTE: Why So Serious? The Importance of Comedy in Tech** - *Chloe Condon, Senior Cloud Advocate, Microsoft*
12:30 PM	1:30 PM	Birds-of-a-Feather Lunch
1:30 PM	2:00 PM	Dessert Break • Visit the EXPO
2:00 PM	3:15 PM	**TMW07** Surviving a Ransomware Attack Using Azure Site Recovery (Demo Fest) - *Émile Cabot & Dave Kawula*
		TMW08 Everything You Need to Know to Manage Modern IT Apps Using Intune - *Peter Daalmans*
		TMW09 Creating and Sharing Awesome PowerShell Modules: A No Nonsense Guide - *Kirk Munro*
3:15 PM	4:00 PM	Networking Break • Visit the EXPO • Expo Raffle @ 3:30 p.m. - Pacifica 7
4:00 PM	5:15 PM	**TMW10** Driving Adoption of the Modern Workplace - *Émile Cabot*
		TMW11 5 Crucial Mistakes Made in IT-Projects (and How to Avoid Them) - *Erwin Derksen*
		TMW12 Life Beyond Functions: Creating Powershell Script Cmdlets (aka Advanced Functions) - *Michael Wiley*
7:30 PM	9:00 PM	Live! 360 Dessert Luau - Wantilan Pavilion
TechMentor Day 3: Thursday, November 21, 2019		
7:30 AM	8:00 AM	Registration • Coffee and Morning Pastries
8:00 AM	9:15 AM	**TMH01** Build Your Azure Infrastructure like a Pro - *Aleksandar Nikolic*
		TMH02 Imposter Syndrome: Overcoming Self-Doubt in Success - *Heather Downing*
		TMH03 Windows Autopilot Deep Dive - *Petri Paavola*
9:30 AM	10:45 AM	**TMH04** How to Get Your Cloud Services Secured with Intune and Azure AD - *Peter Daalmans*
		TMH05 Best Tips to Make the Life of SCCM Admins Easier! - *Panu Saukko*
		TMH06 What I Learned When I Moved my Operations to the Cloud - *Sami Laiho*
11:00 AM	12:00 PM	**TECHMENTOR PANEL DISCUSSION: The Future of IT** *Moderated by Sami Laiho and Dave Kawula; Émile Cabot, Peter Daalmans, Petri Paavola, & Panu Saukko*
12:00 PM	1:00 PM	Lunch - *Oceana Ballroom*
1:00 PM	2:15 PM	**TMH07** Building Real World Labs in Azure - *Dave Kawula*
		TMH08 Implementing Proactive Security in the Cloud - *Sami Laiho*
		TMH09 How to Use PowerShell to Become a Windows Management SuperHero—2019 Edition - *Petri Paavola*
2:30 PM	3:45 PM	**TMH10** Becoming a Community Rockstar - *Cristal Kawula*
		TMH11 Managing Azure VMs with Windows Admin Center - *Aleksandar Nikolic*
		TMH12 Ready, Set, Go: Start Managing Your Windows 10 Devices with Intune - *Panu Saukko*
4:00 PM	5:00 PM	**NEXT? LIVE! 360 NETWORKING EVENT** - *Bradley Ball, Andrew Brust, Ben Curry, Peter Daalmans, Joseph D'Antoni, Marcel de Vries, Billy Hollis, Dave Kawula, Raj Krishnan, Sami Laiho, Thomas La Rock, Vishwas Lele, Rockford Lhotka, Karen Lopez, Matthew McDermott, Agnes Molnar, Brian Randell, Panu Saukko, Jen Stirrup, & Alex Thissen*
TechMentor Post-Conference Workshops: Friday, November 22, 2019		
7:30 AM	8:00 AM	Registration • Coffee and Morning Pastries
8:00 AM	5:00 PM	**TMF01** Workshop: Migrating and Upgrading to Windows Server 2019 - *Dave Kawula*
		TMF02 Workshop: How to Build Intune Managed Windows 10 with the Best User Experience - *Petri Paavola*

Speakers and sessions subject to change

PRODUCED BY CONVERGE360 an 1105 MEDIA company

CONNECT WITH LIVE! 360

twitter.com/live360 @live360

facebook.com Search "Live 360"

instagram.com @live360_events

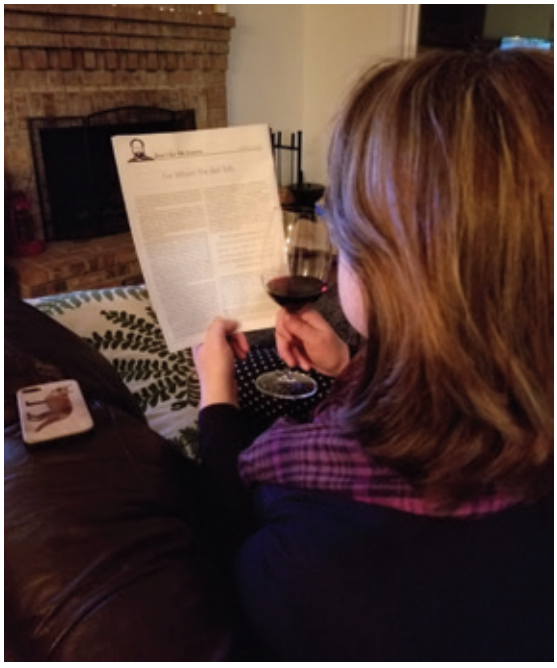
linkedin.com Join the "Live! 360" group!

So Long, and Thanks for All the Fish

My dear friends,

Here we are, sharing our last meeting in this space. I always figured that we'd part sooner, that I'd get fired for shooting off my mouth once too often. But, no, apparently Microsoft had to kill the whole magazine to get rid of me. It feels right, somehow. So, herewith, Plattski's *MSDN Magazine* swan song.

Thanks to my editors, starting with Keith Ward who recruited me for this gig at the end of 2009. He then wisely fled to *Visual Studio Magazine*, inflicting me on poor, unsuspecting Michael Desmond. He put up with me for eight-plus years, from his first day on the job to this final demise. Poor guy, stuck between Microsoft, who didn't want to take *too* much of a public beating (especially in a book it owns), and yours truly, always looking to pour oil on troubled fires for the pleasure of watching the flames. Mike must have really needed the money.



I salute Microsoft for taking its ribbing with good humor, mostly, usually, except when it didn't.

But he's also done a great job on my writing, without taking credit, despite my repeated offers. I'm not letting him weasel out of this one. Good job, Mike, and thanks. And I'm sure your next gig will be far less interesting.

I salute Microsoft for taking its ribbing with good humor, mostly, usually, except when it didn't. These last 10 years have been quite challenging for the company, starting from a world of desktop word processors and spreadsheets, trying and failing to create a phone ecosystem, yet somehow transforming and moving on to leadership in the cloud and AI. It's done well, as its stock price shows, and is poised to continue doing well. (Not quite as well without this magazine, but let that pass.)

Above all, I'd like to thank you, the wonderful people I've met through this column. I knew I'd connected with you back in 2012, when I was teaching in Israel. I showed a picture of my late cat, and said I sometimes let her write my column. A student raised his hand and, "Is that Simba?" To everyone who loved the column and never missed it, like the reader named Clair who sent me a photo of her enjoying it (you can see her in the photo). And to everyone who hated it, but read it anyway, to have something to get angry about: Thank you. Thank you. Thank you. (I mean, hey, it would have sucked doing this all by myself.)

I asked my daughter Annabelle if I should keep this column going. Maybe it's time to put it down? She said, "Daddy, you've got way too much in your head that you need to rant about. If you don't have some place to publish it, you'll probably explode, and I'll have to clean up the pieces." ("Out of the mouths of babes, to fools," as the Good Book says. Or something like that.)

So I've decided to keep my monthly column going on my personal blog, davidsplatt.com. I'll start with the columns that Microsoft would never have allowed, such as the leak I received from Edward Snowden, about Putin's arch-hacker Raskolnikov meddling in the 2016 election.

It's been a fabulous time to be alive and working in this industry. Remember what I told you, and try to at least nudge it in the right direction. And come see me in my new place. You can count on me to keep calling 'em as I see 'em.

Until we meet again: Take care of yourselves. Take care of each other. And take it easy, but take it. L'hitraot. ■

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 have taped down two of his daughter's fingers so she would learn how to count in octal. You can contact him at rollthunder.com.*

Address the ELEPHANT IN THE ROOM

Bad address data costs you money, customers and insight.

Melissa's 30+ years of domain experience in address management, patented fuzzy matching and multi-sourced reference datasets power the global data quality tools you need to keep addresses clean, correct and current. The result? Trusted information that improves customer communication, fraud prevention, predictive analytics, and the bottom line.

- Global Address Verification
- Digital Identity Verification
- Email & Phone Verification
- Location Intelligence
- Single Customer View

+ .NET
+ Microsoft® SSIS
+ Microsoft® Dynamics CRM

See the Elephant in Your Business -

Name it and Tame it!

melissa

www.Melissa.com | 1-800-MELISSA

Free API Trials, Data Quality Audit & Professional Services.

ArcGIS for Developers

A complete mapping and analytics platform

The ArcGIS for Developers program offers a full suite of geospatial developer tools and resources. This program includes the ArcGIS Runtime SDK for .NET which enables developers to add map visualization and geospatial capabilities to native apps built using .NET. This empowers app users to do all things spatial—from simple map display and directions to editing and sharing of geographic data to advanced visual analysis. The ArcGIS Runtime SDK for .NET includes WPF, UWP, iOS, and Android APIs that enable .NET developers to build native mapping apps for Windows, iOS, and Android devices.

ArcGIS Runtime goes beyond basic mapping—it combines the power of the ArcGIS Platform with your data to deliver engaging, interactive, and high-quality mapping experiences how and where you need them. ArcGIS Runtime offers the following benefits to help you build and deploy native applications with powerful spatial capabilities:

High Speed Performance

ArcGIS Runtime is built on a C++ native core and takes advantage of GPU and other device resources to optimize display and interaction with both maps and data.

Work Offline

Use local maps and data to support offline workflows. ArcGIS Runtime supports local access to a wide variety of vector and raster data formats, including GeoPackage, Shapefile, and GeoTIFF. In addition, the ArcGIS Platform provides convenient tools for packaging and delivering maps and their data for offline use.

Advanced Cartography

Create and use beautiful, detailed, interactive symbols to optimize cartographic display, improve map readability, and communicate priorities in both 2D and 3D experiences.

Use the ArcGIS Platform

The ArcGIS Platform provides powerful tools, applications, and services to create, curate, stylize, host and secure your geographic data, maps, and workflows. The ArcGIS Platform also includes ready-to-use content and services you can combine with your data. ArcGIS Runtime works seamlessly to utilize and collaborate with all parts of the ArcGIS Platform to provide a comprehensive mapping solution.

Cross-Platform and Cross-Device

ArcGIS Runtime SDK for .NET integrates with the cross-platform capabilities of the .NET Platform. .NET developers can use ArcGIS Runtime to support mapping experiences and geospatial workflows in a single, shared codebase for native client applications that target Windows, iOS, and Android devices.

Integrated with Visual Studio

ArcGIS Runtime for .NET easily integrates with Visual Studio via NuGet and includes a set of application templates to get started. The SDK includes extensive guide and API reference documentation as well as source code for samples, example apps, and a toolkit to supplement native mapping components and workflows.

Get started at No Cost

Sign up for the ArcGIS Developer Program for free and get access to the ArcGIS Runtime SDK for .NET (and other ArcGIS APIs and SDKs): **https://developers.arcgis.com/sign-up**. Start building powerful mapping apps today!

About Esri

Esri, the global market leader in geographic information system (GIS) software, offers the most powerful mapping and spatial analytics technology available.

For more information, visit → **developers.arcgis.com**