THE MICROSOFT JOURNAL FOR DEVELOPERS

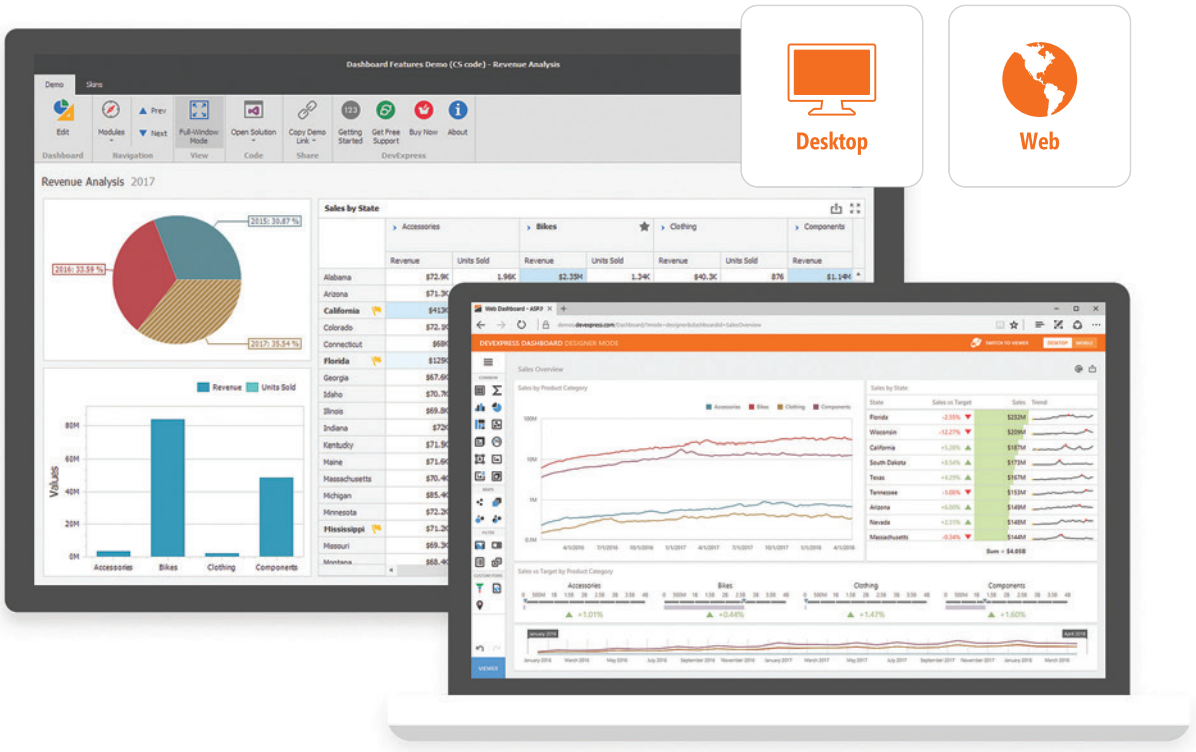
OCTOBER 2019 VOL 34 NO 10

msdn magazine

DevExpress®

DevExpress Dashboard for .NET

Create and distribute royalty-free decision support systems and effortlessly share business intelligence across your entire enterprise.



Data Grid · Pivot Grid · Card View · Charts · Gauges · Treemap · Map · Range Control · Filter Control · Rich Text Editor · Image

With DevExpress Dashboard for .NET, creating insightful and information-rich decision support systems for executives and business users across platforms and devices is a simple matter of selecting the appropriate UI element (Chart, Pivot Table, Data Card, Gauge, TreeMap, Map, Grid or simple Filter elements) 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.

Learn More Today — Try DevExpress Dashboard Risk Free for 30 days

devexpress.com/dashboard

THE MICROSOFT JOURNAL FOR DEVELOPERS

OCTOBER 2019 VOL 34 NO 10

msdn magazine

F# Web Apps with the SAFE Stack.....25

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

OCTOBER 2019 VOLUME 34 NUMBER 10

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

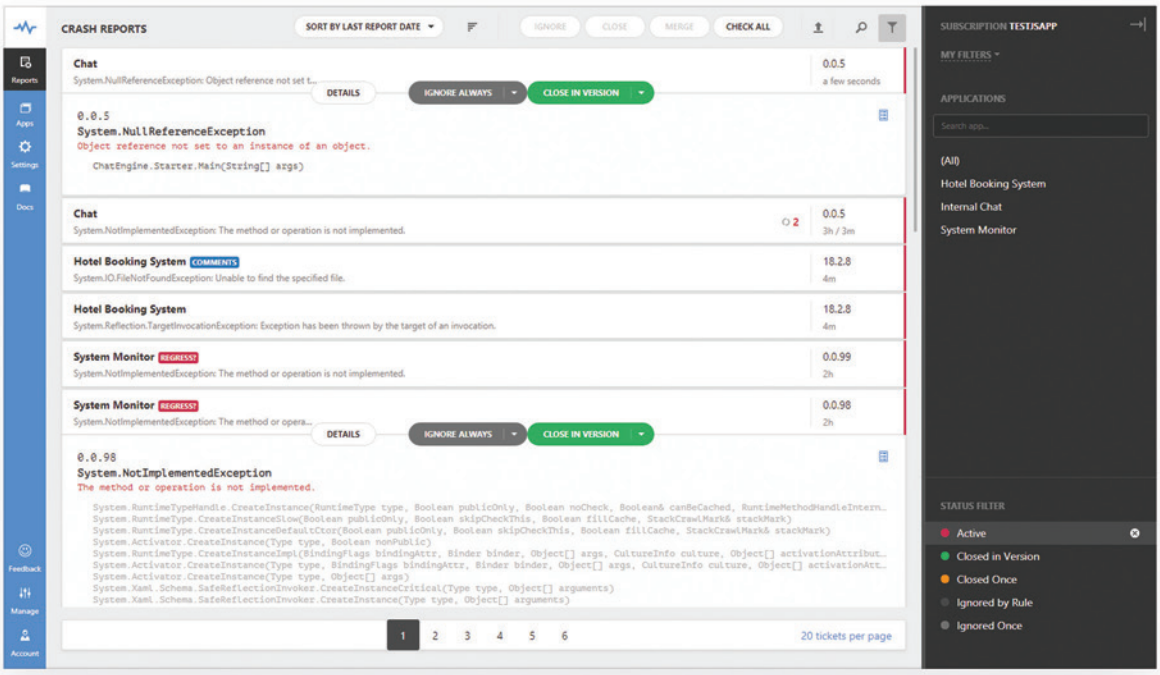
1906 100 2006 AMERICAN BUSINESS MEDIA

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®

MICHAEL DESMOND

And Now a Word from Our Columnists

When *MSDN Magazine* closes shop after the November issue, we'll be saying farewell to a gifted group of longtime columnists who have been the heart and conscience of the magazine. I wanted to take a moment this month to recognize their contributions and consider the lessons they've learned in the process.

I'll start with James McCaffrey and his Test Run column, which got its start in January 2003, after the release of .NET Framework revolutionized the way applications were developed and tested. For the next several years, McCaffrey says, his column explored how to test .NET apps natively, with an emphasis on automation. But a sharp pivot was already in the offing.

"During this time, I was one of the few research engineers who worked on machine learning systems, including 'TMSN,' the predecessor to the ML.NET framework that was released earlier this year," McCaffrey explains. By 2012, breakthroughs in storage, processing and algorithms convinced McCaffrey that machine learning was about to explode. "So, in May 2012 the Test Run column switched focus to machine learning and artificial intelligence with an article titled 'Dive into Neural Networks.'" (msdn.com/magazine/hh975375)

Looking back, McCaffrey singles out two columns: "Use Bee Colony Algorithms to Solve Impossible Problems" (April 2011, msdn.com/magazine/gg983491) and "Artificial Intelligence—Particle Swarm Optimization" (August 2011, msdn.com/magazine/hh335067). He says: "Both articles illustrate the joy of computer science—creating beautiful abstractions of reality that solve practical problems, which can't be tackled using standard approaches."

I asked Dino Esposito what Cutting Edge column stood out to him over his decades with the magazine. He recalled one about the transition from ASP.NET Web Forms 1.1 to 2.0, and a feature called "personalization" that was a built-in, provider-based API to support custom application settings. He showed how developers could achieve the new functionality in the 1.1 version, and in doing so, he says, "basically rewrote for the current version what the team at Microsoft was writing for the next."

Esposito says the accomplishment gave him a "sense of power," and produced an outpouring of feedback from readers excited to extend the existing framework ahead of version 2.0. He says: "Today, it's much harder to do, given the significantly increased complexity of things, but it's a very nice memory and still a goal to try to pursue anyway—to learn and do it yourself."

The Working Programmer columnist Ted Neward wouldn't argue that last point. His 10-part "Multiparadigmatic Design" series (msdn.com/magazine/ff955611) in 2010-2011 stood out to him, he says, "largely because that was the first time I'd really had to bear down and explore, investigate and describe what had always been a much murkier concept floating around in my head."

His conclusion: Penning the column forced clarity of thought. "Writing for the column definitely helped me learn things about subjects I thought I knew, and after writing, knew better."

Julie Lerman says she often faced "steep learning curves and a lot of research" with her Data Points columns, and singles out her look at document databases ("What the Heck Is a Document Database?," msdn.com/magazine/hh547103), which forced her to wrap her head around NoSQL concepts. She also says that returning to previous topics after discovering new solutions could be both immensely rewarding and "a little humiliating." ("Refactoring an ASP.NET 5/EF6 Project and Dependency Injection," msdn.com/magazine/mt632269)

The experience, Lerman says, expanded her horizons as a developer. "I was able to broaden my technical expertise in so many directions—digging Git, Docker, Azure, document databases—as well as follow the path of Entity Framework even to this day. So much fun!"

Speaking of fun, resident bomb thrower David Platt clearly relished his role as back-page instigator at *MSDN Magazine*. And he particularly enjoyed his columns on Visual Basic 6, including the June 2012 piece, "The Silent Majority: Why Visual Basic 6 Still Thrives" (msdn.com/magazine/jj133828), that produced more first-month traffic than almost any article in the magazine over the last 10 years.

"Nothing stirred up a hornet's nest like these columns, both pro and con," Platt says. "Talk about pouring oil on troubled fires!"

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

NEW RELEASE

Intellifront BI | from $23,246.35

Christiansteven

Cost-Effective Data Analytics & Business Intelligence.

- Design and serve visually stunning real-time reports using Grids, Charts, Gauges, Pies & more
- Add automatically updating KPI cards with aggregation, rounding, units, goals, colors & glyphs
- Create & assign Reporting, KPIs and Dashboards as consolidated "Canvases" to user groups
- Give your users a simple and intuitive self-service portal for their BI reports and documents
- Secure User Access with Active Directory Integration, Single Sign On & 2-Factor Authentication

BEST SELLER

DevExpress DXperience 19.1 | from $1,439.99

DevExpress

A comprehensive suite of .NET controls and UI libraries for all major Microsoft dev platforms.

- WinForms – New Sunburst Chart, Office Navigation UX, SVG Office 2019 skins
- WPF – New Gantt control, improved Data Filtering UX and App Theme Designer
- ASP.NET & MVC – New Adaptive Layouts, improved Rich Text Editor and Spreadsheet
- Reporting – Vertical Band support, Free-hand drawing and improved Report wizards
- JavaScript – New HTML/Markdown WYSIWYG editor, Improved Grid and TreeList performance

NEW RELEASE

PBRS (Power BI Reports Scheduler) | from $9,811.51

Christiansteven

Date & time Scheduling for Power BI reports with one Power BI License.

- Exports reports to PDF, Excel, Excel Data, Word, PowerPoint, CSV, JPG, HTML, PNG and ePub
- Send reports to email, printer, Slack, Google Sheets, folder, FTP, DropBox & SharePoint
- Uses database queries to automatically populate report filters, email addresses & body text
- Adds flexibility with custom calendars e.g. 4-4-5, holidays, "nth" day of the month, etc.
- Responds instantly by firing off reports when an event occurs e.g. database record is updated

BEST SELLER

LEADTOOLS Medical Imaging SDKs V20 | from $4,995.00 SRP

LEADTOOLS

Powerful DICOM, PACS, and HL7 functionality.

- Load, save, edit, annotate & display DICOM Data Sets with support for the latest specifications
- High-level PACS Client and Server components and frameworks
- OEM-ready HTML5 Zero-footprint Viewer with 3D rendering support and DICOM Storage Server
- Medical-specific image processing functions for enhancing 16-bit grayscale images
- Native libraries for .NET, C/C++, HTML5, JavaScript, WinRT, iOS, OS X, Android, Linux, & more

We accept purchase orders.
Contact us to apply for a credit account.

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

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

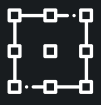
GSA Schedule Contract GS-35F-0188R

The most developer friendly blockchain platform

neo.org

NEO Blockchain Toolkit for .NET

Smart Contract Debugger



NEO Express

Smart Contract Compiler Enhancement

Visual DevTracker

NEO-FX Library

Available at Visual Studio Code Marketplace

NEO Blockchain Toolkit

JULIE LERMAN

Hybrid Database Migrations with EF Core and Flyway

During development, it's common to use EF Core's migration commands to not only create migration files but to also execute migrations directly on your development database. You can apply the migrations with the PowerShell *update-database* command or the command-line interface (CLI): *dotnet ef database update*. What's nice about letting EF Core update the database this way is that you have total control over when the database is migrated to a new schema. And I love having control over things. I can trigger the database migration, then open up the database to verify I got what I expected. But this can only happen during design time. It's an explicit action to perform.

Another path for triggering EF Core to migrate the database using the migration files is to call the Database.Migrate or the Database.EnsureCreated method somewhere in your code. For example, in the first installation of my recent article series, "EF Core in a Docker Containerized App" (msdn.com/magazine/mt833405), I worked out a way to trigger EF Core to migrate my database at application startup. This is handy during development (and for demos, such as the article's code) if you don't want to manually use commands to update your database—especially during early stages of development when your model is evolving rapidly.

But applying migrations at application runtime has some critical drawbacks. A major one is that you're depending on your software to be responsible for the database. Is that what you want? Is that what your DBA wants? Even if you've added custom migration files to handle administrative tasks, it's probably not going to make your DBA happy. What if you have multiple APIs that use the same database? Which one should be responsible for the database?

But there's also a deeper technical problem you're inviting by calling Migrate from within the software. In that earlier article, I was publishing my API into a Docker image. What if I were to start up several containers to serve up that API? Each container would be hitting that Migrate method on startup. While unlikely, it's possible that multiple servers might call Migrate at the same time and try to perform a migration on the database concurrently. This could create unexpected (and detrimental) side effects.

So, when it's time to move from development and QA to production, letting EF Core update the database for you is generally not the best approach. Even the EF Core documentation (bit.ly/2XVQEHy) makes a note about applying migrations at runtime: "While it's great for apps with a local database, most applications will require more robust deployment strategy like generating SQL scripts."

EF Core migrations do let you create SQL scripts, even idempotent scripts (that avoid applying schema modifications that have already been applied). An app like Octopus Deploy can use deployment steps to read EF Core migrations, generate the SQL scripts and then execute them. Or you can generate the SQL yourself using EF Core migration commands (PowerShell: *Script-Migration* or CLI: *dotnet ef migrations script*). You (or the database guru in your org) can tweak these scripts to add more database knowledge into them if needed. Either way, this gives you the ability to then feed them to some other application that's designed to manage your database updates, rather than you having to manage and execute the SQL manually. Redgate's SQL Change Automation (formerly called Ready Roll), for example, is an ace at managing SQL Server updates with your scripts. There are also tools that aren't just for SQL Server, such as Flyway (flywaydb.org), now part of the Redgate stable, or Liquibase (liquibase.org).

During development, it's common to use EF Core's migration commands to not only create migration files but to also execute migrations directly on your development database.

Recently I used a combination of EF Core migrations and Flyway to demonstrate using Docker containers to apply database migrations. This alternative allows integrating the evolution of the database schema into the same automated Continuous Integration/Continuous Deployment (CI/CD) process you use to deploy new versions of your application code. You can see a video of this from my recent NDC Oslo session at youtu.be/BL3TfA0UEr8. Having the database servers hosted in a Docker container for developer and tester machines, as well as in an automated CI/CD pipeline, eliminates the need to install the server anywhere. And I used a separate Docker container for the short period needed to run Flyway, letting it apply any migrations to the database and then shut itself down.

Code download available at msdn.com/magazine/1019magcode.

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

Flyway Basics

The Flyway app describes itself as "version control for your database" and it works with a large number of relational databases. There are three editions: the free Community edition, Pro—which adds more features—and Enterprise. It's pretty straightforward: You provide a series of SQL script files and it will run them for you. Similar to EF migrations, it stores a history of which scripts have been run in a special migrations history table. This table will get stored in whatever default database is indicated by the database connection string. You'll see more about that later.

Flyway is cross-platform and can run on a variety of operating systems—including mobile platforms. I've used it in Docker containers to execute scripts as part of a CI/CD process, as demonstrated in the video I mentioned earlier. Flyway also has a rich API, although I'll only be using the most basic function—migrate—for this article.

The order in which scripts are executed is critical. Flyway has a naming convention to ensure that scripts are executed in the correct order. Key elements of the name are "V" for version (to differentiate from U for Undo scripts and R for Repeatable scripts). I'm going to focus only on Version scripts in this article. Next comes the version number, where you can use an underscore separator for sub versions. The critical double underscore follows, to separate the version info from a descriptive name. I once struggled with SQL files getting ignored because I used a single underscore by mistake in this position, so watch out for this. A typical file name might look like *V1__BuildTables* or *V1_1__CreateRoles*.

I installed Flyway to my MacBook using Homebrew (*brew install flyway*). You can find install instructions for other operating systems on Flyway's website. The brew installation ensured that flyway was on my file path so I had easy access to it from the command line without having to go look for it. Flyway requires Java and brew guided me to easily install what I needed.

Here's how I went about the hybrid migration using EF Core and Flyway.

Start with EF Core Migration Files

I let EF Core create migration files for the model I'm working with. For example, my simple model for the conference session I mentioned is shown in **Figure 1**. There's an Agilista class (to keep track of all of my gal pals in the Agile community) and a Category class to represent the areas of Agile they focus on, such as Agile Testing.

> The order in which scripts are executed is critical.

I then let EF Core create my first migration using *dotnet ef migration add initial*. The resulting file uses the migration API to describe defining the tables, relationship constraints and indexes.

I also want a little bit of test data that I'll seed so I can do a bit of testing (though not performance testing). Therefore, I added the EF Core HasData method in the DbContext class to define some seed data. **Figure 2** shows some of that code, although the downloadable sample code inserts a few more agilistas and categories.

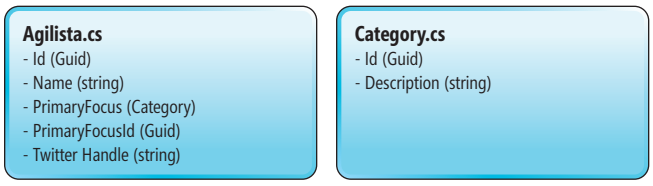


Figure 1 **My Simple Domain Model**

You can read more about HasData in my August 2018 column (msdn.com/magazine/mt829703).

I created a second migration with *dotnet ef migration add seeddata*, which describes inserting the data into the tables created in the first migration.

> Instead, you want to be explicit about which migrations to script; that way, the outputted SQL can represent incremental changes to the database and be applied as needed.

Rather than letting EF Core apply these migrations (which means it will generate the relevant SQL in memory and then execute it), I'll tell EF Core to just output the SQL for me. I'm intentionally excluding the *–idempotent* flag when generating these scripts. Flyway will take care to avoid running scripts more than once. However, EF Core's script command, by default, will script every migration file found. That means if you script your first two migrations, then add a third migration and call script again, a script will be created for all three. This won't work for my plans.

Instead, you want to be explicit about which migrations to script; that way, the SQL that's output can represent incremental changes to the database and be applied as needed. I could combine both of my existing migrations into a single script, but because they perform different types of tasks, I'll create separate scripts for explicitness.

Figure 2 **Seeding Some Data with EF Core HasData**

```
protected override void OnModelCreating (ModelBuilder modelBuilder
{
  modelBuilder.Entity<Category> ().HasData (
    new {Id = new Guid ("167d1f6b-a93d-49e4-8a0d-e651369e018b"),
         Description = "Agile Testing" },
    new {Id = new Guid ("5f6d6f80-9f9a-469e-9036-07ecbb3971ea"),
         Description = "Exploratory Testing" }
    );

    modelBuilder.Entity<Agilista> ().HasData (
      new { Id = new Guid ("5efdb55d-1205-419f-8a0b-9cc7a15f8565"),
            Name = "Lisa Crispin",
            PrimaryFocusId = new Guid ("167d1f6b-a93d-49e4-8a0d-e651369e018b") },
      new { Id = new Guid ("83eda86f-c652-4666-ba17-db90b218a54b"),
            Name = "Linda Rising",
            PrimaryFocusId = new Guid ("c5b6a0e8-e43f-4765-906f-e15e019a19d8") }
    );
```

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

The PowerShell syntax uses the -To and -From parameters. From specifies the last run migration and therefore doesn't include it in the generated script:

```
script-migration -To initial -Output createAgilistaCategoryTables.sql
script-migration -From initial -To seeddata -Output SeedData.sql
```

The CLI syntax doesn't use the parameter names. You only supply the names of the from and to migrations. While these aren't required arguments, if you do want to specify which migrations to script, you must provide both arguments. Therefore, if you're starting from the beginning, you need to use the number zero in the from position as I'm doing in the first of these two CLI commands:

```
dotnet ef migrations script 0 initial -o createAgilistaCategoryTables.sql
dotnet ef migrations script initial seeddata -o SeedData.sql
```

You can see the details of these files in the accompanying download.

Transforming the Scripts for Flyway

There are some facets of the new script files that are specific to EF Core. Because of that, I need to do a little extra work to use them outside of EF Core migrations. There are five tasks:

- Create the database
- Remove EF Core migrations-specific logic
- Ensure the database is targeted prior to running SQL
- Rename the files to follow Flyway naming conventions
- Move the files into a sub-folder called sql

Creating the Database

The first step is to ensure the database gets created. Why? Although the EF Core Migrations Update Database command creates a database if needed, that logic is internal to the API, not expressed in the migration files. So, you'll need SQL that creates the database before the new scripts are run, and perhaps more tasks to be performed on the database server or the database itself, such as creating roles or users. I want to create a login, user and role on the SQL Server and then, after creating the new database, set up the same login, user and role in the database.

I've created another SQL file, InitDatabase.sql, to perform all of those tasks and, again, because of space constraints, you'll need to grab the download to see the details. However, key in the file is the command to create the database:

```
CREATE DATABASE DB_Agilistas
```

Removing EF Core-Specific SQL

The second task I need to perform is to remove EF Core migrations-specific logic from the generated SQL scripts. That's all of the references to the Migrations_History table that EF Core maintains. You won't need these because you won't be using EF Core to keep track of the migrations.

The first script file (createAgilistaCategoryTables.sql) contains SQL to create the table and to update it after the migration has been performed.

At the top of the file you'll see:

```
IF OBJECT_ID(N'[__EFMigrationsHistory]') IS NULL
BEGIN
  CREATE TABLE [__EFMigrationsHistory] (
    [MigrationId] nvarchar(150) NOT NULL,
    [ProductVersion] nvarchar(32) NOT NULL,
    CONSTRAINT [PK___EFMigrationsHistory] PRIMARY KEY ([MigrationId])
  );
END;
```

This needs to be removed, as well as any commands to insert data into that table (which you'll find at the end of each SQL file). For example:

```
INSERT INTO [__EFMigrationsHistory] ([MigrationId], [ProductVersion])
VALUES (N'20190418172940_seeddata', N'2.2.3-servicing-35854');
```

Specifying the Correct Database

EF Core knows which database to execute migrations on. But your output script files don't contain this information. Be sure to include it at the top of each of the SQL files, although not the one that's creating the database. That way you don't have to worry about the connection string defaulting to a different database such as master. I'm adding the following to the top of the two generated SQL files:

```
USE DB_Agilistas
GO
```

Renaming the Files for Flyway

Finally, you'll need to be sure the file names align with Flyway convention so they'll be run in the proper order. Note again that I'm focusing only on forward-movement files that start with the letter V for Version, and not Flyway's ability to have Undo or Repeatable scripts.

Currently my three files are named:

- InitDatabase.sql
- createAgilistaCategoryTables.sql
- SeedData.sql

So, you'll need SQL that creates the database before the new scripts are run, and perhaps more tasks to be performed on the database server or the database itself, such as creating roles or users.

To ensure that Flyway will execute them in the proper order, I renamed them:

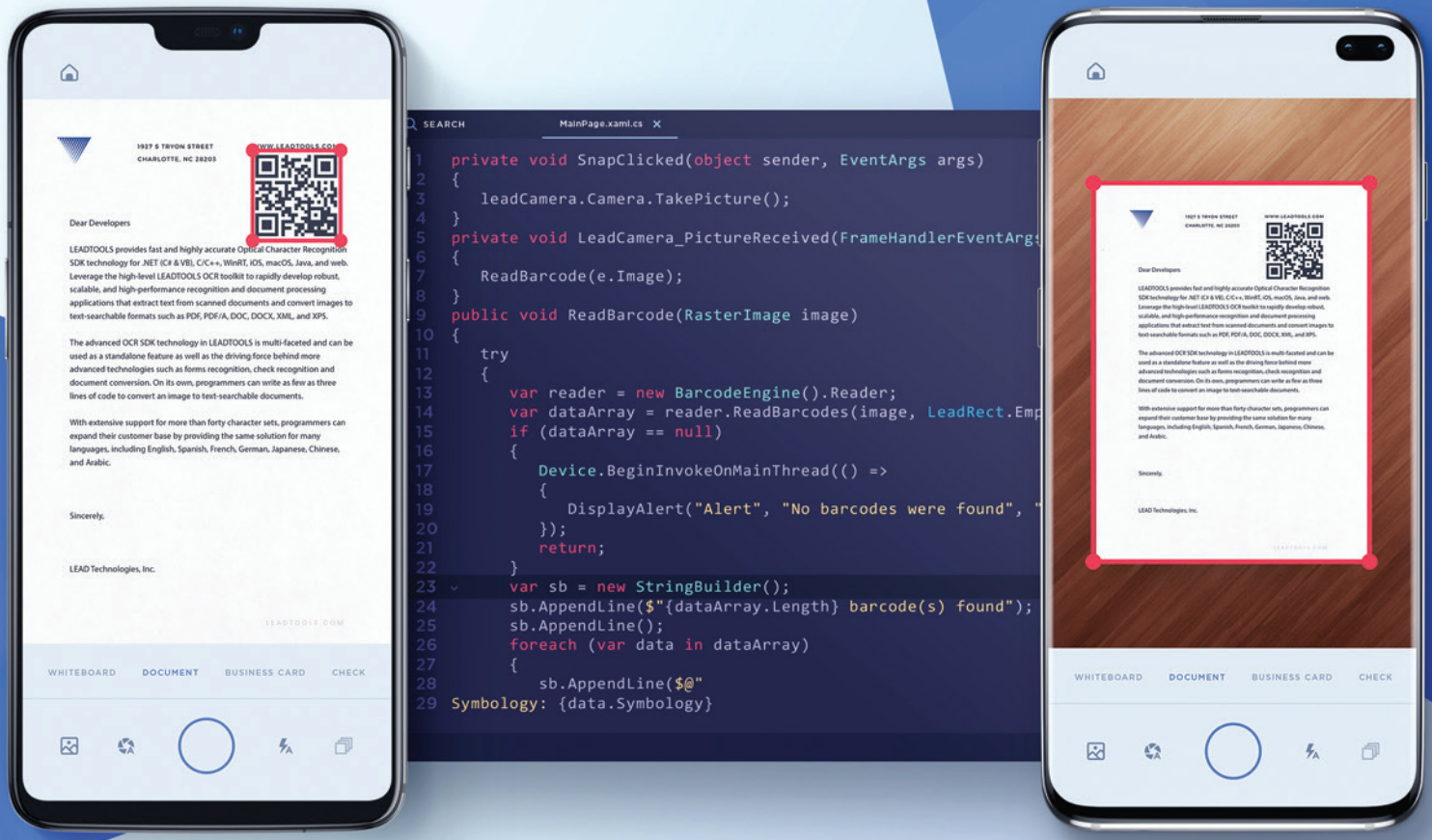
- V1_0__InitDatabase.sql
- V1_1__createAgilistaCategoryTables.sql
- V1_2__SeedData.sql

They all start with V1 because they're part of my first pass. In addition to one last reminder about the importance of that double underscore, I'll admit it took me a number of tries before I realized that the name of one of the files (which Flyway kept ignoring) began with a lowercase v rather than upper.

After making all of these changes, I moved the three SQL files into a sub-folder in my project called sql, which follows a convention of Flyway you can override if needed.

LEADTOOLS®

XAMARIN CAMERA SDK



The LEADTOOLS Xamarin Camera Control provides a common camera interface with full control over Android and iOS camera functions. With the LEADTOOLS Xamarin Camera Control, there is no need to write native code for each device platform. One line of XAML markup opens the door to capture images, record video, control the flash/torch, and focus the camera. Additionally, developers can tap into the live camera feed to process frames in real-time or add LEADTOOLS features, including barcode recognition, OCR, and image processing filters.

OCR | BARCODE | PDF | OFFICE FORMATS | ANNOTATIONS | VIEWERS | + MANY MORE

Get Started Today

DOWNLOAD OUR FREE EVALUATION

LEADTOOLS.COM

installed_rank	version	description	type	script	checksum
1	1.0	init	SQL	V1_0__init.sql	-1722142353
2	1.1	createAgilistaCategoryTab…	SQL	V1_1__createAgilistaCateg…	-462858239

Figure 3 **The Flyway_schema_history Table Shows Two Migrations Have Been Applied**

migrations are inserted as rows in the flyway_schema_history table. **Figure 3** shows part of the listing of the two new rows.

Had there been any problems, Flyway would have stopped executing and output details about the issue. As I'm not doing anything advanced in this scenario, the migrations aren't being run in a transaction. Therefore, at this early stage of learning, you'd need to roll back any changes that were made before any failures.

Now let's introduce the database change represented by the third SQL file. I'll change the name back to V1_2__SeedData.sql so Flyway will see it.

I'll rerun *flyway migrate* and this time, the logs tell me it found three migrations, noted that the schema version was 1.1 and that it applied the seedData migration. Here's some of the detail from the ouput:

```
Successfully validated 3 migrations
Current version of schema [dbo]: 1.1
Migrating schema [dbo] to version 1.2 - seedData
WARNING: DB: Changed database context to 'DB_Agilistas'.
Successfully applied 1 migration to schema [dbo]
```

Looking at the database in Azure Data Studio, I can see all of the assets created by the various migrations, including the new tables and their data. The flyway_schema_history table also has a new row representing the latest migration.

Running Flyway from the Command Line

I already had a SQL Server running in a Docker container that was exposed to my environment on localhost:1601. (There are now many articles in the Data Points series that use SQL Server in Docker. The initial introductory article from July 2017 can be found at msdn.com/magazine/mt784660.)

> Rather than including the URL, user, password and other needed parameters in every command, I've stored them in a Flyway configuration file, flyway.conf, in my project.

With Flyway, you need to specify the database connection with a URL. I'm using the syntax for SQL Server and setting master as the default database:

```
jdbc:sqlserver://localhost:1601;database=master
```

The Flyway docs provide examples for various database types. Rather than including the URL, user, password and other needed parameters in every command to Flyway, I've stored them in a Flyway configuration file, flyway.conf, in my project:

```
flyway.url=jdbc:sqlserver://localhost:1601;database=master
flyway.user=sa
flyway.password=P@ssword1
flyway.locations=filesystem:sql/
flyway.mixed=true
```

The last configuration—mixed—allows both transactional and non-transactional SQL commands in the same migration file.

Because I want you to see Flyway perform an update, I'm going to muck up the name of the third SQL file, making the initial letter a lowercase "v"—v1_2__SeedData.sql—causing Flyway to ignore it on the first run.

With this in place, I can finally run *flyway migrate* at the command line in my project folder.

Flyway will read the configuration file and then look to see if its migration table, named flyway_schema_history, exists on the server yet. In my case it doesn't, so Flyway creates that table inside the default database (master). Then it notes how many migration files it found, compares those files to what's in the flyway table (none yet) and runs any that aren't yet listed in the table.

Flyway's logs note the current version of the schema, which is "<Empty>" at this point; then that it "successfully validated 2 migrations"; and that it "successfully applied 2 migrations." The applied

A More Robust Path for Server Migrations

Flyway uses concepts similar to those you may be familiar with from working with EF or EF Core, especially with respect to the history table in the database server. However, using an explicit mechanism to perform migrations on a production database rather than depending on EF Core to do them from within your application is preferable when your deployment has any level of complexity.

There's still more to learn if you plan to use Flyway in a hybrid approach to leverage EF Core migrations for certain tasks and to perform migrations on your production database. But my goal here was to introduce you to the basic concept and let you get a feel for how the pieces go together. And while I chose Flyway as the tool for running my SQL scripts, you might want to translate this hybrid approach to some other tool that allows you to run scripts. Remember, though, that one advantage of Flyway (and Liquibase) is that it works with a number of relational databases. Other tools focus on a single database such as SQL Server. ■

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 technical experts for reviewing this article: *Julia Hayward (Redgate), Diego Vega (Microsoft)*

Manipulating Documents?

APIs to view, convert, annotate, compare, sign, assemble and search documents in your applications.

Try GroupDocs APIs for FREE

Download a Free Trial at

https://downloads.groupdocs.com

GROUPDOCS

GroupDocs.Viewer

View over 50 documents and image formats in any application using document viewer APIs.

GroupDocs.Annotation

Add annotations to specific words, phrases and any region of the document.

GroupDocs.Conversion

Fast batch document conversion APIs for any .NET, Java or Cloud app.

GroupDocs.Comparison

Compare two documents and get a difference summary report.

GroupDocs.Signature

Digitally sign Microsoft Word, Excel, PowerPoint and PDF documents.

GroupDocs.Assembly

Document automation APIs to create reports from templates and various data sources.

GroupDocs.Metadata

Organize documents with metadata within any cross platform application.

GroupDocs.Search

Transform your document search process for advance full text search capability.

- GroupDocs.Text
- GroupDocs.Editor
- GroupDocs.Parser
- GroupDocs.Watermark

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

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!

Save $300 when you register by October 25!

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!

TED NEWARD

Python: Flow Control

In the last article (msdn.com/magazine/mt833510), I took a start down the path toward Python, getting it installed and paying homage to the Gods of Computer Science. Obviously, there's not much you can do at that point, so it's time to dive deeper into Python by examining its flow control structures, like if/then statements, loops and exceptions.

One thing before I begin, though: I never really touched on the subject of editors. While most readers of this magazine will be accustomed to using Visual Studio for all editing purposes, building solutions and projects and such, Python doesn't really fit that model. And certainly, while you could use Visual Studio simply as a text editor, it's generally pretty much a heavyweight for that sort of use. (As my videogaming sons like to put it, "There's no kill like OVERkill.") As a result, Visual Studio Code is likely to be the editor of choice for most folks. However, the Anaconda distribution that I discussed last time (which ships as part of the Visual Studio installer) comes with another option, a more IDE-ish experience called Spyder. It's certainly not as extensive as Visual Studio is, but it makes for a nice "halfway point" between the Visual Studio Code editor-first experience and the Visual Studio IDE-everything experience. (For those who are fans of the JetBrains suite of tools, it also makes PyCharm, which has a certain amount of popularity among the Python crowd.) As always, the choice is up to you—most Python projects have no dependencies on any particular editor or environment.

Armed thusly with your editor of choice, let's proceed.

Branching

The most basic branching construct is the classic if/then statement, which does exactly the same thing as in pretty much every other programming language ever invented: Test an expression and if the test yields a true result, execute a block of code; if the result is false, either bail entirely or execute an alternative "else" branch. Python adds only one wrinkle to this, which brings me to an important distinction of Python, that of *significant whitespace*. In Python, you don't denote scope blocks using punctuation characters (like the "{" and "}" in C-family languages such as C#), you simply indent a number of characters, typically four, like so:

```
# If-then
name = "Zoot"
if name == "Zoot":
  print("Naughty, naughty Zoot!")
elif name == "Sir Robin":
  print("Brave, brave Sir Robin...")
else:
  print("I have no idea who you are")
```

(Note that I may reduce the indentation in some code listings here to two characters for spacing in the magazine; but stylistically, Python is super-sensitive to whitespace and prefers four, so you should stick with that.) Take particular care to note the lack of parentheses around the expression, and the colon at the end of each test expression—the colon will be an indicator to you that Python expects the next line or sequence of lines to be indented. As a matter of fact, if you remove the "print" line from either of those blocks, and leave nothing in their place, Python will complain quite vigorously. If you don't want code in an expected indented block, Python provides the "pass" statement, or a standalone "…," to indicate a no-op statement:

```
# If-then
name = "Zoot"
if name == "Zoot":
  pass
elif name == "Sir Robin":
  ...
else:
  print("I have no idea who you are")
```

When this is executed, nothing will be printed, because it matches on the first expression but then passes on actually doing anything.

> To loop, you have two constructs, *while* and *for*.

It's also important to note that the Python mantra—"There's only one way to do it"—holds here, as well, with respect to this sort of sequential testing of a value. Where other languages, like C#, Java, Ruby or F#, have some form of multiple-choice decision-making construct (usually denoted by the keyword "switch"), Python chooses instead to stick with the conceptually simpler "if-then-elif" collection. In other words, Python has no "switch" because a whole list of "elif"s can serve the same purpose. (Whether this is an advantage or drawback is open to debate and interpretation, of course.)

Looping

To loop, you have two constructs, *while* and *for*. While will continue to loop and execute a block until a test condition is determined to be false, and for iterates across a *sequence*, which is a generic way of referring to a variety of "iterable things." The syntax for a while loop is pretty straightforward, with one interesting quirk that also applies to for. Have a look:

Visual Studio LIVE!

EXPERT SOLUTIONS FOR ENTERPRISE DEVELOPERS

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!

Visual Studio Live! is your road map to the Microsoft Development world, featuring code-filled days, networking nights, and intense developer training. Whether you are a developer, software architect, or a designer, Visual Studio Live!'s multi-track events include focused, cutting-edge education on the Microsoft and .NET platform to keep you and your business competitive and future-ready.

SUPPORTED BY

Microsoft | Visual Studio | msdn magazine | Visual Studio MAGAZINE

PRODUCED BY

CONVERGE 360 — An 1105 MEDIA company

vslive.com #VSLIVE

```
# While
count = 0
while count < 5:
  print("Hello")
  count = count + 1
else:
  print("Done saying hello")
```

Spot the quirk? As you might infer, the "else" block gets executed after the while loop ceases looping, regardless of how many times the loop was actually executed. Even if the count variable had started at 6, the else block would've fired. This is something Python offers that no other mainstream language does, and it provides a useful way to provide cleanup or other sorts of behavior that should always be executed, regardless of what actually happens with the loop.

The for loop operates similarly, but operates against a sequence, so let's start with a list, which fundamentally is an array, but is dynamically resizable:

```
# For
numbers = [1, 2, 3, 5]
for n in numbers:
  if n == 3:
    continue
  print(n)
else:
  print("3! From the Book of Armaments")
```

As might be expected, this will recite King Arthur's count down for the Holy Hand Grenade of Antioch.

Using a list might be acceptable for small numbers, but obviously if you want to count up to 100, having to construct a list of 100 elements ahead of time would be impractical. In some other languages, you have a particular flavor of the for-loop construct to set a starting point, ending point and iteration, but remember, in Python, there's only one way to do it. (On top of which, that three-part for-loop construct is pretty cumbersome when you look back on it.) For these scenarios, Python provides a function, called range, to provide a sequence of numbers suitable for use in a loop:

```
ns = range(0,100)
for n in ns:
  print(n)
```

As you might infer, the "else" block gets executed after the while loop ceases looping, regardless of how many times the loop was actually executed.

Running this snippet will also demonstrate that the results of the range function are bottom-inclusive, top-exclusive (meaning you'd see zerio, but not 100). For ranges that want to "step" by more than one, you can construct the range to jump by a set value by passing a third parameter to the range:

```
ns = range(0,100, 5)
for n in ns:
  print(n)
```

This will print the values five, 10, 15 and so on.

Regardless of the source of the iteration, the for-loop construct allows for manipulation of the loops via the *break* and *continue* keywords; the former arbitrarily exits the loop, and the latter causes the loop to begin again. Note as well that the range doesn't have to start low and grow high; you can just as easily start with zero and "grow downward" by stepping by negative one each time, if so desired:

```
ns = range(0,-100,-1)
for n in ns:
  print(n)
```

For the most part, any experienced object-oriented developer familiar with the language constructs of Java, C#, C++ or any of their kin will find Python's flow-control constructs to be easily digestible.

But the real power of the range function comes into play when you realize that the underlying construct that the for-loop is operating off of is a *sequence*: an object that knows how to produce the next value expected. Many people will assume that this is similar to the iterator object that's frequently used as part of C# (the IEnumerator interface), but in fact, this is a little bit more powerful than that. If you were to print the returned object from the range function, what you'd see isn't the full list of values, but the object itself; that is to say, this snippet of code:

```
ns = range(0,10)
print(ns)
```

This code will print "range(0,10)," not "[0,1,2,3,4,5,6,7,8,9]," because no list has been constructed. The sequence object knows its lower range (0), its upper range (10), and how to produce each next successive value (add one to the current value) as requested, so it doesn't need to construct the entire collection. This means the sequence object occupies the same amount of memory whether it's a sequence from zero to 10 or from zero to 10 million. (This is often referred to as a "stream" in other languages.) Python makes use of sequences in a variety of ways throughout the platform, not just in for-loop constructs.

Exceptions

Finally, Python also supports exception-handling similar to the way C#, Java and C++ do: To signal an error condition that can be trapped and handled further up the call stack, Python allows you to "raise" an object instance (or a class, which Python will take to mean create an instance and then raise it) and immediately exit the current function, in the same way that you "throw" exceptions in C#:

```
raise Exception()
```

This will throw a rarely used generic exception type. I'll talk more about how to create new exception types and so on after I cover how to create classes (and subclasses—anything "raise'd must inherit from a base class).

More to the point, Python allows you to catch these exceptions in much the same way that C#, C++ and Java do, by using the "try" keyword to establish a guarded block of code that, if an exception is raised, will transfer control to one of a series of "except" blocks, with a "finally" clause that will be guaranteed to fire regardless of how control exits the guarded block. What's new is that Python also allows an "else" clause off of the "try" block, which fires in the event no exception is generated:

```
# Try/Catch
try:
  print("Let's do bad math")
  impossible = 10 / 0
except ZeroDivisionError:
  print("Can't divide by zero, man")
else:
  print("That worked?!?")
finally:
  print("Finally clause")
```

There are a few forms of exceptions that allow for catching multiple exception types and so on, but the core form is the same. As with C#/CLR exceptions, if the exception passes outside of the top-level function in the currently executing program, the thread will terminate and print a (hopefully helpful) message to the console. Exception types are objects, and custom exception types can be constructed for more precise error-handling logic in more complex programs, but doing so requires understanding how to construct objects in Python, which is still a bit out-of-scope for what I've covered so far.

Wrapping Up

For the most part, any experienced object-oriented developer familiar with the language constructs of Java, C#, C++ or any of their kin will find Python's flow-control constructs to be easily digestible. If anything, Python's insistence on rigid conformance to the "there's only one way to do it" mantra makes it that much easier to bring the language into play; there's no trying to defend when to use "switch/case" instead of a number of "if/else-if" statements, or trying to justify the use of a "do-while" against a "while." In many respects, this makes Python a much easier language to take on as a first language, which is likely why Python is rapidly becoming the first language for many new programmers, as well as for data scientists who don't want to program but "ju,st let me do stuff with my data."

We're hardly done here, though; Python, like any mainstream programming language, supports the idea of capturing behavior into a named construct—the function—and that's where we'll turn our collective heads next. In the meantime … 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*

Exploring the tidyverse

In my last article (msdn.com/magazine/mt833459), I explored the fundamentals of the R programming language, which is widely used in the data science space. At the end of the article, I pointed out that all the code written for the article was in "base R." While base R is capable of loading, exploring and visualizing data, it's not the only way to perform data analysis in R. At the end of the article, I briefly mentioned the tidyverse (tidyverse.org), a collection of packages for R that align to common design principles and are designed to work together seamlessly. Package developers that would like to add to the tidyverse must adhere to the tidyverse style guide (style.tidyverse.org). This enables a consistent experience for developers and ease of interoperability between packages.

Tidyverse packages are designed to simplify and streamline the data science process of load, prep, train and visualize by providing a more consistent development experience across the various libraries. A good analogy would be how jQuery simplified Web development by creating a more consistent programming surface across the DOM, event handling and more. While not a language per se, jQuery made JavaScript more productive by lessening the friction of their most common tasks.

The tidyverse libraries are open source and available on GitHub (github.com/tidyverse). The core tidyverse modules include packages needed for everyday data analyses and exploration. As of tidyverse 1.2.0, the following packages are included in the core distribution: ggplot2, dplyr, tidyr, readr, purrr, tibble, stringr and forcats. Dozens of other useful packages are also included in the tidyverse, but aren't loaded automatically with library(tidyverse). See tidyverse.org/packages for details. This article will explore the basics of how to load, filter and visualize data the "tidyverse way."

Recall that last month's column used post count data from my blog as a sample dataset. This dataset is simple with enough variation to demonstrate the power and ease of tidyverse packages. It also helps to use the same sample dataset to facilitate comparisons between base R and tidyverse R.

Loading Data with readr

The readr package provides a fast and easy way to read rectangular data files, such as .csv files. It can flexibly parse many types of data files, while handling errors robustly. To get started, create a new R language Jupyter Notebook. For details on Jupyter Notebooks, refer to my February 2018 article on the topic at msdn.com/magazine/mt829269. In the first blank cell, enter the following code to load the .csv file data and display it:

```
library(readr)
fwposts <- read_csv("franksworldposts.csv")
fwposts
```

Note that above the tabular output with the contents of the .CSV file is text that highlights how each record was parsed and that the output is a tibble with 183 rows and four columns. Base R uses data frames to store tabular data. In the tidyverse, a tibble is the equivalent structure. In fact, tibbles are data frames, but they modify some default data frame behaviors to meet the needs of modern data analytics. More information about tibbles can be found at tibble.tidyverse.org and in-depth documentation resides at r4ds.had.co.nz/tibbles.html.

Look at the following message:

```
Parsed with column specification:
cols(
  Month = col_character(),
  Posts = col_integer(),
  `Days in Month` = col_integer(),
  PPD = col_double()
)
```

While the read_csv function did properly load and parse the data, it didn't automatically detect that the Month column was a date field and labeled it a character field instead. However, I would like to preserve this data type in the schema. To do this, I need to pass along a col_types parameter to the read_csv function that explicitly defines the column schema. As readr correctly guessed all the column data types except one, I can use the existing schema as a guide.

> The readr package provides a fast and easy way to read rectangular data files, such as .csv files.

To see the current schema or specification of the tibble, enter the following code into a blank cell and execute it:

```
spec(fwposts)
```

Enter the following code, which takes the original inferred schema and adjusts how the Month column is parsed. The "%b-%y" format string matches the format of the column with the three-letter month abbreviation and a two-digit year separated by a dash, like so:

```
fwposts <- read_csv("franksworldposts.csv",
  col_types = cols(
   Month = col_date(format = "%b-%y"),
   Posts = col_integer(),
   `Days in Month` = col_integer(),
   PPD = col_double()
  )
)

fwposts
```

Note that the Month column is now properly marked as a date field in the schema.

Filter and Manipulate Data with dplyr

The dplyr package provides a consistent set of functions for nearly all data manipulation and querying tasks. To use the dplyr package, enter the following code into a new cell and execute it:

```
library(dplyr)
```

> The dplyr package provides a consistent set of functions for nearly all data manipulation and querying tasks.

With the dplyr package loaded, I will use it to view only the months with 100 or more posts by using the pipe operator %>% to pass the tibble to the filter method. Enter the following code into a new cell and execute it:

```
fwposts %>%
  filter(Posts >= 100)
```

The output shows only the months with 100 or more posts.

Pipes

Pipes are a fundamental concept to the tidyverse. They're used to emphasize a series of actions where the item to the left of the pipe operator becomes the input to the right of the pipe operator. Software developers familiar with fluent style of coding will immediately recognize this pattern. To view only those rows with 100 posts or more in ascending order based on the Posts column, I would write the following code:

```
fwposts %>%
  filter(Posts >= 100) %>%
    arrange(desc(Posts))
```

To see them in descending order, I would add a call to the desc method, like this:

```
fwposts %>%
  filter(Posts >= 100) %>%
    arrange(desc(Posts))
```

I could further analyze the data by adding an additional pipe to a summarize function. Summarize functions create one value summarizing the values in a table. Enter the following code to view the number of rows, the mean post count and the mean PPD values, like so:

```
fwposts %>%
  arrange(desc(Posts)) %>%
    summarize(n(), mean(Posts), mean(PPD))
```

The values returned will be 183, 36.94536 and 1.215148.

Working with Groups

Note that the summarize function returned one value for the entire dataset. If I wanted to track how the values changed over time, I could group the values by year. To do this, I'll import a new library (lubridate) to extract the year from the Month column. The lubridate library makes working date values easier. Using dplyr's mutate method, I will add a new column named Year to store the extracted value. The following code does just that, assigns it to the fwposts variable and displays it. Take note of the new column:

```
library(lubridate)

fwposts <- fwposts %>%
  mutate(Year = lubridate::year(Month))

fwposts
```

Now that the Year column has been added, I can use group by it and display a summary based on the group. Enter the following code into a new cell and execute it:

```
posts_by_year_summary = fwposts %>%
  group_by(Year) %>%
    summarize(PostCount=n(), AvgPosts = mean(Posts), AvgPPD = mean(PPD))
posts_by_year_summary
```

Note that there are now summary rows for each year and that the columns have names. However, the PostCount column contains the number of rows in a given year, not the sum of the posts. To change this, I'll need to use the sum function to add up the values in the Posts column. Enter the following code into a new cell and execute it:

```
posts_by_year_summary = fwposts %>%
  group_by(Year) %>%
    summarize(Records= n(), PostCount=sum(Posts), AvgPosts = mean(Posts),
AvgPPD = mean(PPD))
posts_by_year_summary
```

Now I have the total number of posts for the year, in addition to the number of rows for a given year, stored in posts_by_year_summary. If I wanted to remove all columns except for the Year and PostCount, I'd use the select function to choose only the fields I wanted to keep. Here's the code:

```
year_postcount_only <- select(posts_by_year_summary, Year, PostCount)
year_postcount_only
```

> I can use the arrange and desc methods to sort the values to find the year with the highest posts.

Alternatively, I could use the select function to remove columns. Execute the following code (the contents of the year_postcount_only tibble should be identical):

```
year_postcount_only <- select(posts_by_year_summary, -c(Records, AvgPosts, AvgPPD))
year_postcount_only
```

Just as before, I can use the arrange and desc methods to sort the values to find the year with the highest posts. Enter the following code into a blank cell and execute it, like this:

```
year_postcount_only %>%
  arrange(desc(PostCount))
```

The results show that 2017 was a busy year on the blog. The next step would be to plot these values onto a graph and explore the data visually.

Visualization with ggplot2

Fortunately, the ggplot2 package makes creating graphs from data straightforward, as it allows for creating graphics declaratively. Simply provide the data and instructions on mapping data columns to graphic elements, as well as which graph type to employ, and ggplot2 handles the rendering. For instance, to create a scatter plot of PostCount by Year, enter the following code to generate the graph as seen in **Figure 1**.

```
library(ggplot2)
ggplot(year_postcount_only, aes(Year, PostCount) ) + geom_point()
```

To connect the points on the graph with a line, enter the following code into a new cell and execute it, like this:

```
ggplot(year_postcount_only, aes(Year, PostCount) ) + geom_line() + geom_point()
```

ggplot2 also provides rich formatting options. Enter the following code to create a more colorful version of the line:

```
ggplot(year_postcount_only, aes(Year, PostCount) ) + geom_line(linetype="dashed",
  color="blue", size=1) + geom_point(color="red", size=2)
```

To further explore the data, I can generate a histogram to explore the distribution of the data. For example, I want to get an idea of the distribution of how many posts there have been across all 16 years. Enter the following code to use data from the fwposts tibble to build out a histogram:

```
ggplot(fwposts, aes(Posts) ) + geom_histogram()
```

As the graph shows, most months have 50 posts or less, with one very noticeable outlier. In statistical terms, the number of posts is skewed right. To get some finer granularity, I will set the binwidth to 10. Enter the following code and run it to create the graph as shown in **Figure 2**:

```
ggplot(fwposts, aes(Posts) ) + geom_histogram(binwidth=10)
```

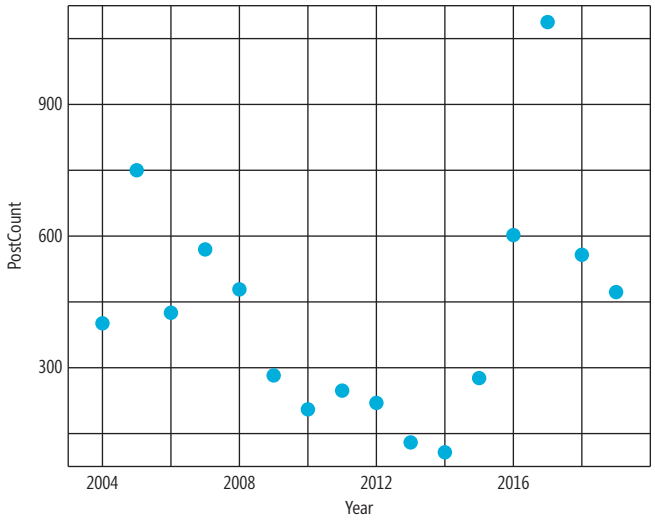


Figure 1 **Scatter Plot of PostCount by Year as Rendered by ggplot2**

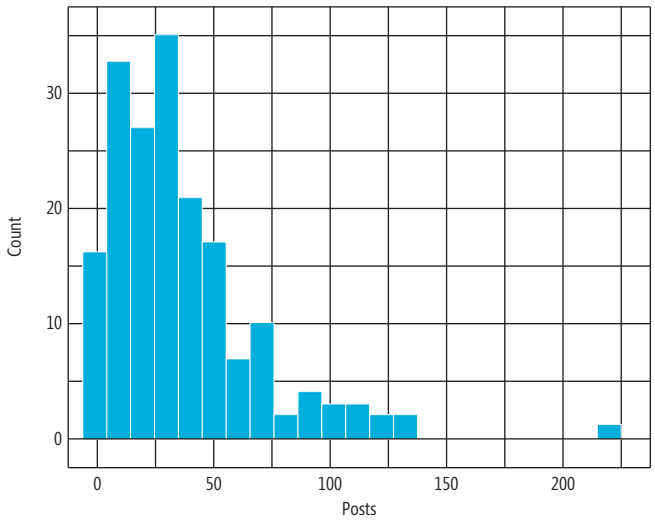


Figure 2 **Histogram of Posts per Month with a Binwidth of 10**

> While base R is perfectly acceptable for most data science-related tasks, many R developers prefer to use the tidyverse suite of libraries for increased productivity.

The histogram in **Figure 2** shows that the most common number of posts lies between 30 and 40. Adjusting the binwidth to lower values increases the granularity.

Another useful visualization for understanding distribution of numeric values is the box plot. A box plot is a standardized way of displaying the distribution of data based on a five-number summary: the minimum value, first quartile, median, third quartile and maximum value. Fortunately, generating a box plot is simple in ggplot2. Enter the following code and execute it to see the box plot for Posts:

```
ggplot(fwposts, aes(x=Posts, y=Posts)) + geom_boxplot()
```

The generated plot shows that the first and third quartile are between around 13 and 50, with a number of outliers at or above 100. For more information about box plots, read this excellent article on the topic: bit.ly/2IbqkmX.

Wrapping Up

While base R is perfectly acceptable for most data science-related tasks, many R developers prefer to use the tidyverse suite of libraries for increased productivity. In this article, I walked through the most common steps in a typical data science pipeline: loading, exploring, manipulating and visualizing data.

These open source package libraries provide a developer experience optimized for data science. The fluent style of programming provides better code readability, streamlined workflow, and a consistent experience across multiple libraries. In fact, there's even a style guide for package developers to follow so that new libraries fit nicely into the tidyverse. ■

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: *David Smith*

SAFE Stack: Functional Web Programming for .NET Core

Isaac Abraham

As a longtime .NET developer who's spent years in the C# world, I was used to working with shrink-wrapped Web frameworks designed for C#, such as the ubiquitous ASP.NET MVC and Web API frameworks.

When I started using F# a few years ago, I gained access to a flexible, powerful and pragmatic language that ran on a framework I already knew, and opened the door to a much more open and dynamic ecosystem than I was used to. However, working with Web applications was an uncomfortable experience: The server side meant either using ASP.NET, and trying to shoehorn F# into an object-oriented (OO) framework—or giving up ASP.NET completely and using a totally different Web server. Meanwhile, working on the client invariably meant using JavaScript, throwing the type system away completely and needing to upskill to deal with its unique "features." Moreover, deploying F# applications into Azure had no support from Microsoft and little or no documentation on how to do this yourself.

The SAFE Stack changes all that. This open source stack allows you to write F# Web apps without needing to fit your code into an OO framework or library or require you to be an expert in CSS or HTML to create rich, compelling client-side Web applications. The default SAFE stack is made up of the following four components:

- **Saturn:** A lightweight F# library that sits on top of ASP.NET Core to provide a high-performance and reliable Web server.
- **Azure:** Microsoft's cloud platform offering, used for hosting and platform services.
- **Fable:** A library that allows seamless interop between F# and JavaScript in the browser.
- **Elmish:** A library that allows you to write UIs that fit into functional programming (FP) best practices rather than, for example, MVVM or MVC.

Let's start by looking at the SAFE stack from a programming model point of view, looking at the three core libraries.

This article discusses:

- Functional programming basics
- SAFE Stack components
- Web development
- Full stack applications

Technologies discussed:

F#, SAFE Stack, .NET Core

Code download available at:

safe-stack.github.io/docs

Why F#?

Your first reaction on reading this might be: "Why bother with F#?" Or, perhaps, "F# for Web programming? I thought F# was just for math and science!" The truth is, F# is a general-purpose programming language that runs on .NET and .NET Core—just like C#—and can work with the entire NuGet ecosystem of libraries, with excellent interop with C#.

The difference is that where C# has language features that encourage a style of programming that leads to stateful, object-oriented, imperative and statement-based programming, F# leads to writing stateless, function-oriented, declarative and expression-based applications.

So while you can implement the exact same types of applications in both languages, such as line-of-business Web applications that use a SQL Server database, or highly scalable back-end services hosted in Azure, the way you "join the dots" and orchestrate calls and errors to .NET methods, NuGet packages and external services is different.

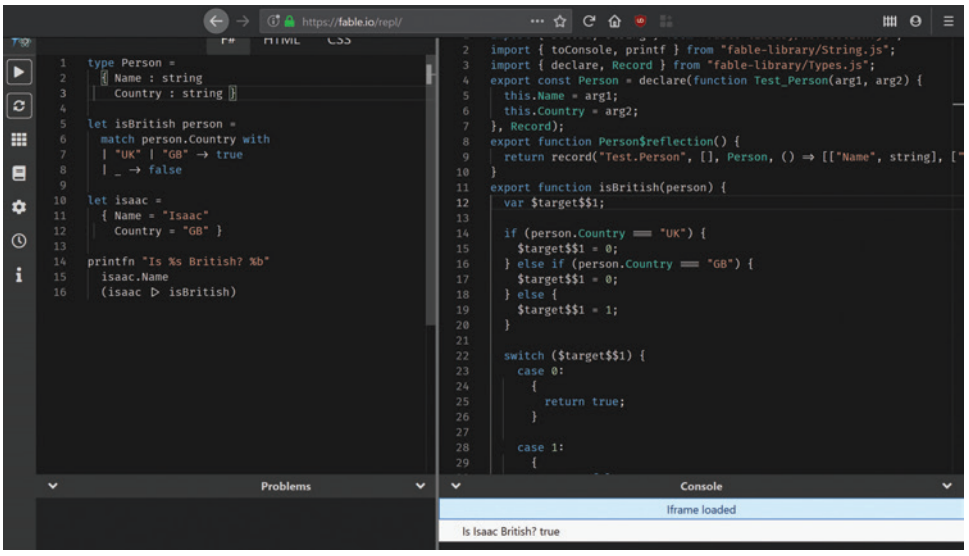


Figure 1 **The Fable REPL**

Indeed, C# already has some FP features, primarily those introduced in C#3 that revolve around LINQ—and that's just the beginning. There's a good reason FP-esque features are cropping up in many languages that have OO roots, because they lead to fewer bugs and easier-to-read code. So, if you enjoy using tuples, switch expressions and non-nullability, F# will almost certainly be a better fit—they're the "default" way of writing code.

Enough about F#. Let's start by looking at the SAFE stack from a programming model point of view, looking at the three core libraries.

Saturn: Functional Programming on the Server

As you probably know, especially in today's world of distributed applications, microservices and containers, writing stateless Web applications often leads to simpler, more scalable systems. In fact, if you think about Web programming fundamentals, you'll see that the entire HTTP model is actually akin to a simple stateless function definition: an input over HTTP—such as a GET request to a specific URI—which returns an HTTP response code and some payload—for example, 200 and some JSON data; there's nothing intrinsically OO about it.

Of course, when you look at ASP.NET, you often think of things like controller classes, dependency injection and so on, but ultimately those are just layers wrapped on top of this simple model. Saturn takes a different approach to a standard ASP.NET application by creating a lightweight programming paradigm that allows you to directly tie HTTP *endpoints* into *functions that return data*—in other words, matching the HTTP input/output model.

Here's a look at the "Hello, world!" for Saturn. I can fit the entire program into a single code snippet because Saturn wraps all the ASP.NET boilerplate for you and lets you focus on routing and business logic:

```
let myRoutes = router {
  get "/api/foo" (text "Hello, world!")
}

let app = application {
  use_router myRoutes
}

run app
```

As you can see, Saturn hooks up GET requests to the "/api/foo" endpoint to a code block that simply returns the text, "Hello, world!". The nice thing about Saturn is that while it doesn't force you to get involved with the whole ASP.NET pipeline, it doesn't stop you from doing it, either. If fact, you can easily access the ASP.NET context object and seamlessly make use of all of the ASP.NET Core-compatible NuGet packages.

Saturn makes use of some F# smarts to create a succinct DSL for writing and composing routes such as *get* and *post* functions that map to the equivalent HTTP verbs, and uses simple helper functions like text, json and xml to quickly create results that set the appropriate values on the HTTP response.

Here's a sample that illustrates how simple routing can be using several additional powerful features of Saturn and F#:

```
let loadOrderDetails (customerId:int) =
  {| OrderCount = 10
     CustomerName = "Fred Smith" |} // loading real data elided...

let orderRoutes = router {
  getf "/customer/%i" (loadOrderDetails >> json)
  put "/customer" saveOrderDetails // etc.
}

let topLevelRoutes = router {
  get "/api/foo" (text "Hello, world!")
  forward "/api/orders" orderRoutes
}
```

Here are some of the points to note from this code snippet:

Route Composition: orderRoutes is "composed" into topLevelRoutes using the forward keyword, so that all routes in orderRoutes are automatically prefixed with /api/orders.

Type safe parameterization: Using F# string placeholder support, %i indicates a route should supply a number, such as api/orders/customer/10, to load orders for customer 10.

Seamless function composition: Using the >> (compose) operator, I "plug together" the loadOrderDetails and json functions to return a record with the order count and customer name, before converting it into JSON.

When you combine Saturn's more powerful features, such as controllers, with the powerful F# type system and excellent data

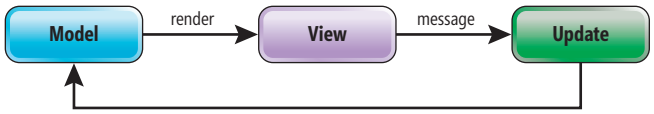


Figure 2 **The Model-View-Update Pattern**

capabilities, such as type providers, on top of the rock-solid ASP.NET Core platform, you can create high-performing, reliable, data-centric Web APIs incredibly quickly.

Life in the Browser with Fable

The browser is where it's at these days, and you'd be hard pushed to find any so-called "full-stack" developer roles that don't require proficiency in JavaScript, which, despite its many shortcomings, is now the de facto standard for client-side Web programming. Fable is SAFE's gateway to writing and executing standard F# in the browser without leaving the JavaScript ecosystem.

Why would you want to do this? Because F# lets you take full advantage of its powerful features—from a compiler that can trap types of errors that many other languages can't, to providing a simple-to-use functional programming model—whilst still letting you take advantage of JavaScript libraries and tools.

There's even a page on the Fable Web site (fable.io/repl) that lets you experiment and see the JavaScript that gets generated by Fable from F# as you type (see **Figure 1**)!

Fable is much more than just a simple "transpiler" for F# to JavaScript: It allows you to "hook into" the standard JavaScript ecosystem using well-known tools such as Babel and WebPack to write standard F# that can interact with the JavaScript world. Rather than a closed framework or environment, Fable is more like a gateway into the JavaScript world. Fable doesn't try to reinvent the wheel—for example, it doesn't actually emit JS itself, but uses the well-known Babel tool to do this—meaning that the emitted JavaScript is high-quality and a known quantity. The JS world has a huge ecosystem and set of libraries on top of which, just as Saturn does with the ASP.NET world, you can interact with using idiomatic F#.

Unlike, for example, Web Assembly, which is more of a "sandbox" inside the browser, Fable applications live *inside* the JavaScript ecosystem, so you can naturally use any existing third-party JavaScript library out there from F#. In effect, Fable applications aren't F# applications running on .NET and the Common Language Runtime (CLR), they're F# applications running in the browser, using NPM packages and Web tools in the JavaScript ecosystem.

Fable also comes out-of-the-box with type-safe wrappers around core HTML and Web concepts (such as DOM, Web storage and geolocation), as well as a set of ready-made wrappers for common JavaScript libraries. You can also make your own; Fable adds F# extensions that make interop simple, and writing simple F# wrappers is quick and easy to do—or you can generate them from TypeScript DefinitelyTyped packages.

Figure 3 **A Simple MVU Application**

```
/// Your data model
type Model =
  { Counter : int }

/// The different types of messages
type Msg =
  | Increment

/// Given the current model and a message, give back an updated model
let update msg model =
  match msg with
  | Increment -> { Counter = model.Counter + 1 }

/// Generates the view for the supplied model
let view model dispatch =
  div [ Class "main-container" ] [
    span [ ] [ str (sprintf "Current value: %i" model.Counter) ]
    button [ OnClick (fun e -> dispatch Increment) ] [ str "Increment" ]
  ]
```

Elmish and the MVU Pattern

Even the JavaScript integration of Fable isn't enough to write complex Web applications, however. What's needed is some sort of "pattern" to write applications that developers can follow. Libraries and frameworks such as AngularJS, Vue and React provide such patterns for JavaScript developers. Unfortunately, most of them are heavily influenced by OO patterns, which means FP developers often have to shoehorn their code into a different way of coding.

Thankfully, there's a better way. The Elm programming language came up with a great FP-friendly paradigm for writing UIs, which has since been adopted in the F# world as the Model-View-Update (or MVU) pattern. There are now .NET "Elmish" libraries for Web, Windows Presentation Foundation, console and (thanks to Microsoft's excellent Fabulous package) even Xamarin applications. **Figure 2** illustrates the MVU pattern at a high level:

Let's see how this translates into code! **Figure 3** shows a simple MVU application that allows you to press a button to increment a counter displayed on the page.

There are a few moving parts here, so let's take it step-by-step:

You define a "model": This is a representation of the state of your application; all the data you want to display or work with in the application goes here. In F#, this is best achieved using a *record* (think of this as a POCO, but as a first-class citizen of the language). In this case, I'm recording the current counter value to display.

You define the "messages" that can occur: Here, I just define one—the event that will fire whenever someone clicks the button to increment the counter value. In F#, instead of using class-based

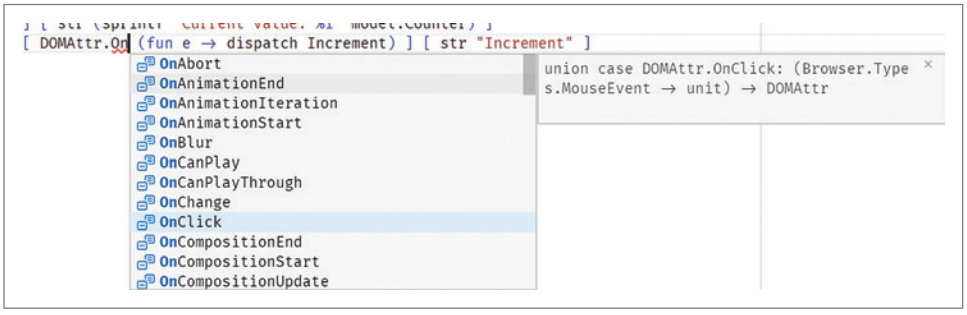
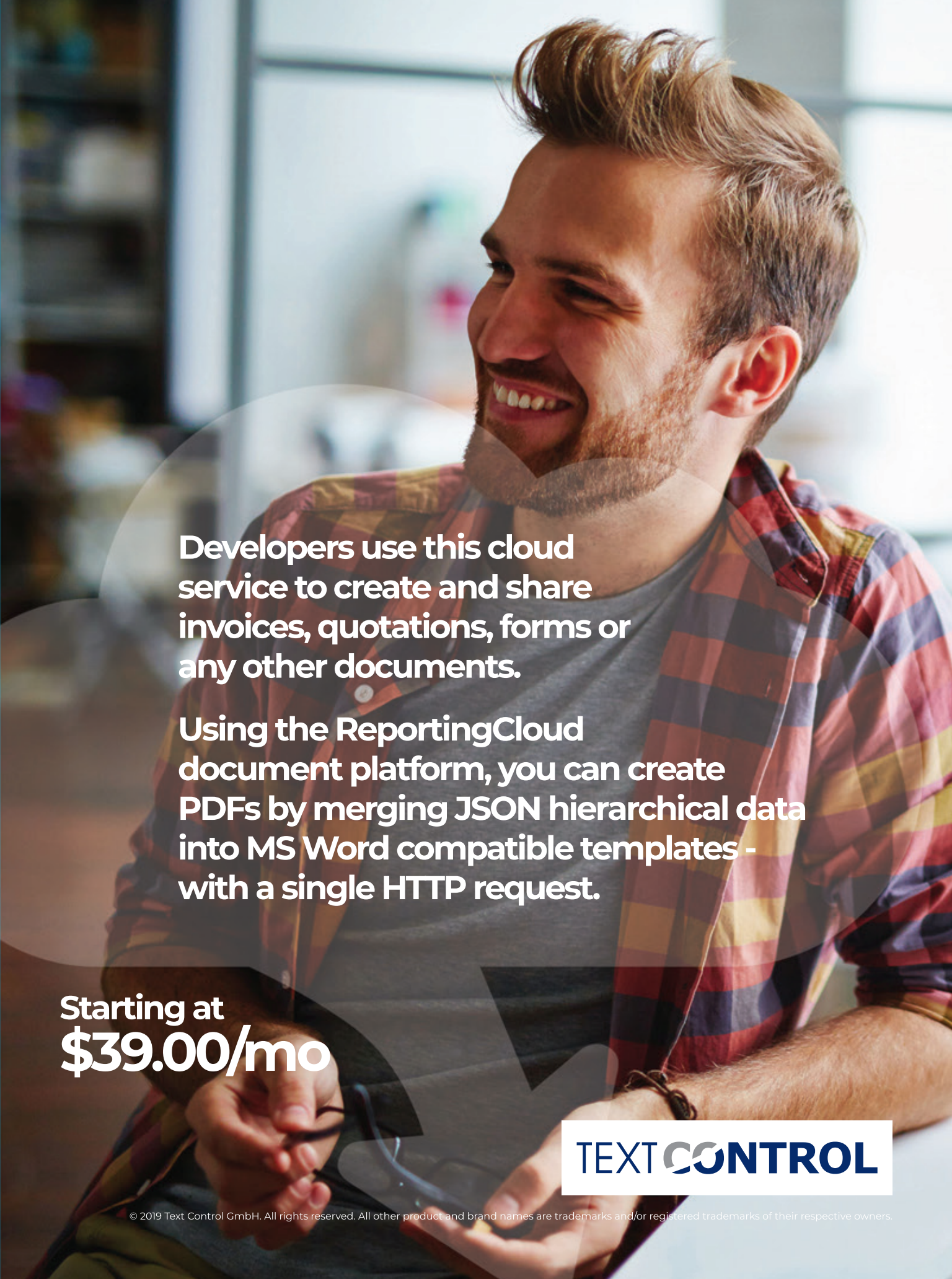


Figure 4 **Type Safe IntelliSense for the Browser**



inheritance, you use a *discriminated union* to model the "is-a" relationship. Every time you create a new message type, you add it to the Message type.

You handle all messages: The *update* function is responsible for taking the current model and a single message, and giving back a new version of the model with any changes applied. Note that you don't mutate the existing model; you copy and update the existing model with a new version, which fits perfectly with FP standard practice. You also use pattern matching to safely "unwrap" the message into the *Increment* message type.

You design your View: The *view* function takes in the current model, and creates some form of "view" that will be rendered to the screen. In this case, SAFE uses a sweet library that provides a strongly typed DSL over the React framework.

Let's dig a little deeper into Elmish now, by taking note of some additional interesting things in the previous sample Elmish.

Strongly typed: As you can see in **Figure 4**, functions and types for React elements such as div and input, and events such as OnClick, are all strongly typed with full IntelliSense.

Lightweight syntax: F# can easily represent a hierarchical view of HTML-like elements. For example, F# has language support for lists—two bracket characters "[ ]" represent a generic list of elements.

Handling user input: The use of the dispatch function allows you to send messages from the UI back into the update function, completing the MVU loop.

Using React: Aside from being a rock-solid and performant library created by Facebook, one of the reasons Elmish sits on top of React is that it provides a highly efficient "diff engine" that can rapidly compare two "virtual" views and update just the HTML elements that have been modified. It's this diff engine that means Elmish can use immutable models and views, but still have excellent performance.

A Word on Pattern Matching

C# 8.0 has introduced a variant of pattern matching with switch expressions. It's a pretty good implementation, although it doesn't support one of the key features that F# discriminated unions give, called *exhaustive* matching. For example, if I were to define a new Decrement message to reduce the counter value, the compiler would immediately tell me where I needed to "fill in" the blanks until it was sure that I had accounted for all possible messages, as shown in **Figure 5**.

This kind of exhaustive matching is one of the reasons F# is such a fun and enjoyable experience—it really helps you to do the "right" thing from the get-go, meaning much less time fighting in the debugger later on.

```
type Model =
    { Counter : int }
val msg : Msg

Full name: msg

Assembly: Client

Incomplete pattern matches on this expression. For example, the
value 'Decrement' may indicate a case not covered by the
pattern(s). F# Compiler(25)

Peek Problem   Quick Fix...

    match msg with
    | Increment → { Counter = model.Counter + 1 }
```

Figure 5 **Exhaustive Pattern Matching in Action**

Value Adds with SAFE

Now that you've seen the different components that make up the default SAFE stack, let's look at some of the other benefits that might not be apparent yet.

Code sharing: Because Fable does such an excellent job writing JavaScript from F#, you can easily share code between client and server—from simple data types to entire modules and functions. For example, you can seamlessly share data contracts between client and server (rather than, for example, JSON or TS types and C# classes) and perform common validation on the client and server, as well as start to make more complex client applications without worrying about increased complexity.

Cloud support: The SAFE stack has great support for different cloud hosting providers. For example, on the Azure front there's built-in support for Azure deployments through ARM templates,

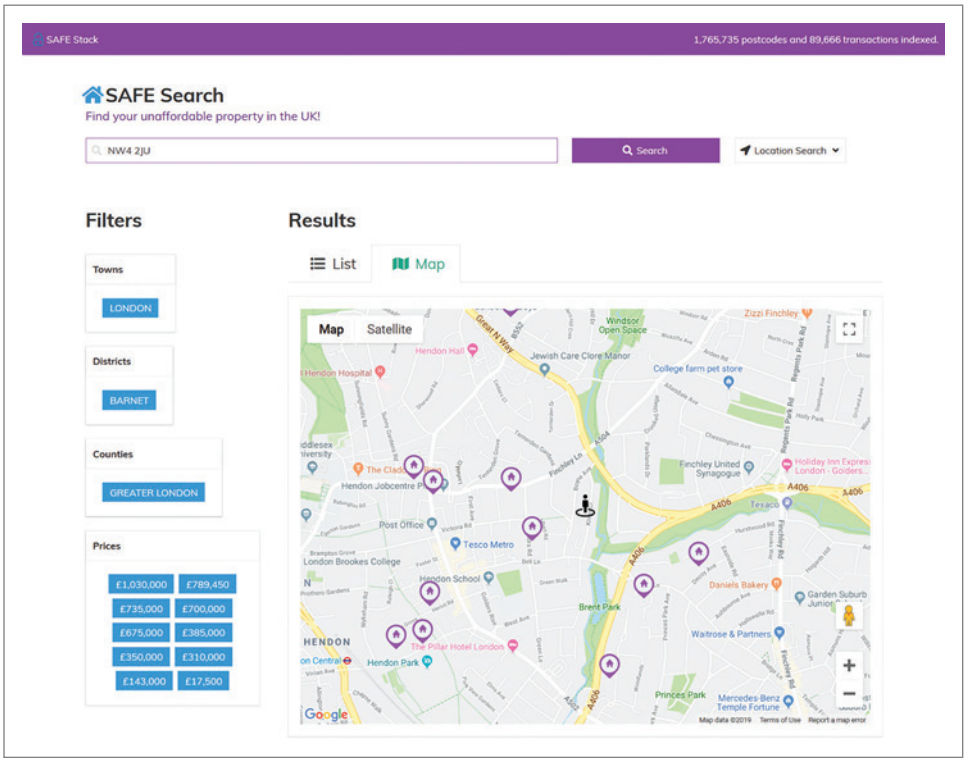


Figure 6 **A Rich Data-Driven Application Using Azure Services**

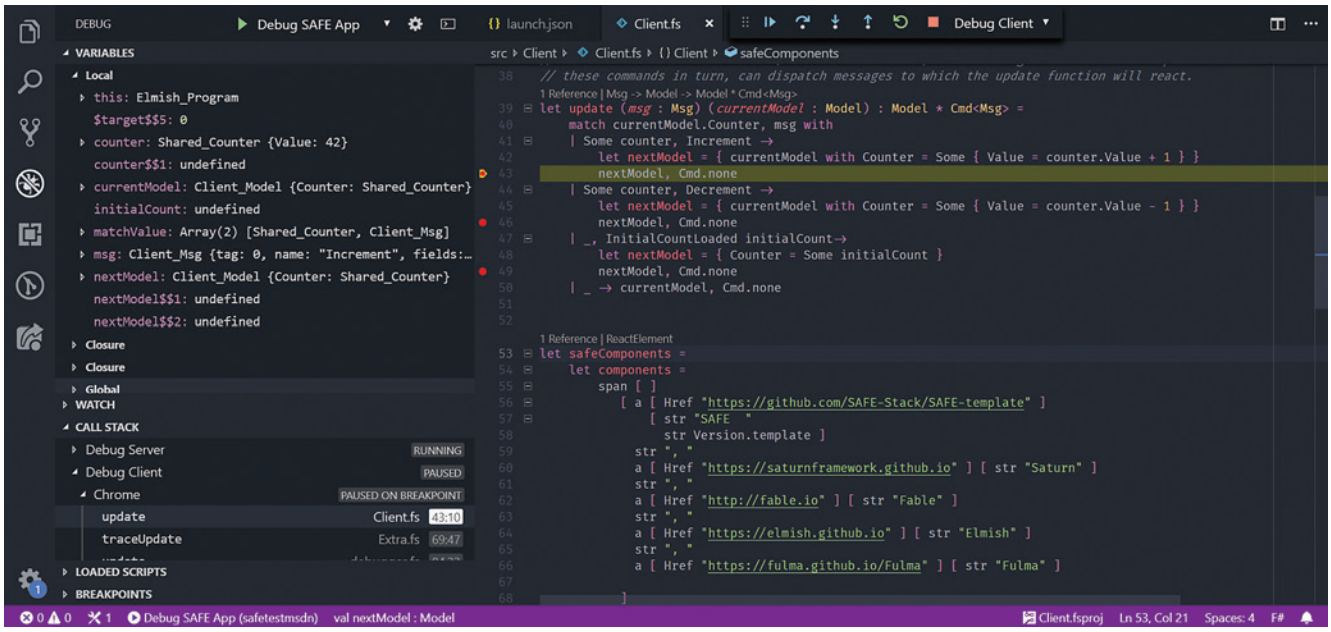


Figure 7 **Unified Client and Server Debugging**

as well as an Azure Storage-type provider that makes access to your Azure Storage data incredibly easy. If you want Docker or GCP support, that's included out-of-the-box, as well. **Figure 6** shows a sample search application using Azure Search and Storage and is hosted on the Azure App Service.

Extensible model: SAFE is more than just a bundled "package" of different technologies; there's a growing ecosystem around it, including libraries that allow seamless client-server interaction through contracts, reactive push-based messaging libraries and WebSocket abstractions.

> SAFE is more than just a bundled "package" of different technologies; there's a growing ecosystem around it.

Debugging support: SAFE changes how you work with F# Web applications. On the server you can use tools like dotnet watch to provide a rapid loop for code changes, but client apps go one step further and take advantage of hot module reloading to allow you to make changes to running applications without restarting, at all! And, thanks to some of the smarts of Fable and the JavaScript community, you can even debug both the client *and* server of your application directly from a single IDE—setting breakpoints and viewing stack traces across both .NET and JavaScript runtimes, both in F#.

Getting Started with SAFE

How do you get started? Here are a few options to get going:

The SAFE Template If you want to dive right in and create a new "empty" SAFE app today, there's a full dotnet project template that has all the bells and whistles, such as generation of ARM templates for Azure, Docker support and so on. There are a couple of prerequisites (bit.ly/2Z8fg0k), but once they're taken care of you can create a new SAFE app like so:

```
dotnet new -i SAFE.Template
dotnet new SAFE
```

Open the folder with VS Code, go to the Debug panel, select Debug SAFE App and press F5. After a short delay while all the JavaScript dependencies are downloaded, a basic SAFE application will start in a Chrome instance.

You can now experiment with the application, creating breakpoints using F9 or clicking in the gutter. Check out how in **Figure 7** the counter value in the Local panel shows the value of the counter (42).

Try the SAFE Dojo You'll find an introductory "dojo" exercise at bit.ly/2TNcixh that introduces you to the different components, starting from a slightly larger "shell" of an application and a set of exercises that you need to complete. It's takes around 60 to 90 minutes and is well worth doing.

Find out More There's also a comprehensive set of documentation and guidance on the SAFE homepage, safe-stack.github.io/docs. From there, you can find user group talks in an area near you; contact the team through social media; and ask questions or reach out to one of the professional organizations that offer SAFE Stack training and consulting.

I hope this article has you excited and interested in trying out both F# and the SAFE Stack—have fun! ■

Isaac Abraham *is a .NET developer, trainer and the director of Compositional IT. He's also a Microsoft MVP for his contributions to functional programming and the cloud.*

Thanks to the following Microsoft technical expert for reviewing this article: *Phillip Carter*

AGENDA-AT-A-GLANCE

#LIVE360

DevOps in the Spotlight | Database Development | Developing New Experiences | Delivery and Deployment | .NET Core and More | Full Stack Web Development

START TIME	END TIME				
Visual Studio Live! 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	**VSS01** Hands-On Lab: Xamarin—Beyond the Basics - *Marcel de Vries & Roy Cornelissen*	**VSS02** Hands-On Lab: Busy Developer's Guide to React - *Ted Neward*	**VSS03** Hands-On Lab: Building Modern ASP.NET Core Web Apps the Right Way with Azure DevOps (Day 1) - *Philip Japikse & Brian Randell*	
2:00 PM	7:00 PM	Pre-Conference Registration - Royal Pacific Resort Conference Center			
Visual Studio 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	**VSM01** Workshop: Developing Modern Web Apps with Azure - *Eric D. Boyd*	**VSM02** Workshop: A Tour of Visual Studio 2019 - *Jason Bock*	**VSS03** Hands-On Lab: Building Modern ASP.NET Core Web Apps the Right Way with Azure DevOps (Day 2) - *Philip Japikse & Brian Randell*	
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			
Visual Studio Live! Day 1: Tuesday, November 19, 2019					
7:00 AM	8:00 AM	Registration • Coffee and Morning Pastries			
8:00 AM	9:00 AM	**VISUAL STUDIO LIVE! KEYNOTE: To Be Announced** - *Chloe Condon, Senior Cloud Advocate, Microsoft*			
9:15 AM	10:30 AM	**VST01** Moving to ASP.NET 2.X — Core Hands - *Philip Japikse*	**VST02** A .NET Developer's Introduction to Unity for 3D Apps & Games- *Nick Landry*	**VST03** How Microsoft Does DevOps - *Paul Hacker*	**VST04** What's New in .NET Core 3.0 - *Jason Bock*
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	**VST05** Fast Focus: Hybrid Web Frameworks - *Allen Conway*	**VST06** Fast Focus: What is Infrastructure as Code - *Paul Hacker*	**VST07** Fast Focus: What's New in EF Core 3 - *Jim Wooley*	
2:00 PM	2:20 PM	**VST08** Fast Focus: Progressive Web Apps (PWA) Empowering Cloud-based Web Applications for Devices - *Andreas Erben*	**VST08** Fast Focus: What is WSL and Why Do I Care? - *Brian Randell*	**VST10** Fast Focus: Git Basics - *Mickey Gousset*	
2:20 PM	2:45 PM	Networking Break • Visit the EXPO - Pacifica 7			
2:45 PM	4:00 PM	**VST11** What's New in .NET Core 3 For Web Developers - *Philip Japikse*	**VST12** Blazing the Web—Building Web Applications in C# - *Jason Bock*	**VST13** Cosmos DB Applications: Part I - *Leonard Lobel*	**VST14** To Be Announced
4:15 PM	5:30 PM	**VST15** Getting Pushy with SignalR and Reactive Extensions - *Jim Wooley*	**VST16** When "We are down" is Not Good Enough. Site Reliability Engineering (SRE) in Azure - *René van Osnabrugge*	**VST17** Cosmos DB Applications: Part II - *Leonard Lobel*	**VST18** What's New in C#8 - *Jason Bock*
5:30 PM	7:30 PM	Exhibitor Reception - Pacifica 7			
Visual Studio Live! Day 2: Wednesday, November 20, 2019					
7:30 AM	8:00 AM	Registration • Coffee and Morning Pastries			
8:00 AM	9:15 AM	**VSW01** Migrating from AngularJS to Angular + TypeScript - *Allen Conway*	**VSW02** AR/VR/Mixed Reality and HoloLens—Making it Real and Useful - *Andreas Erben*	**VSW03** Scrum Under a Waterfall - *Benjamin Day*	**VSW04** Getting Started with Entity Framework Core - *Jim Wooley*
9:30 AM	10:45 AM	**VSW05** TypeScript: Moving Beyond the Basics - *Allen Conway*	**VSW06** Busy Developer's Guide to Flutter - *Ted Neward*	**VSW07** Get Comfortable with .NET Core and the CLI - *Jeremy Clark*	**VSW08** To Be Announced
10:45 AM	11:30 AM	Networking Break • Visit the EXPO - *Pacifica 7*			
11:30 AM	12:30 PM	**LIVE! 360 KEYNOTE: To Be Announced**			
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	**VSW09** Extending ASP.NET Core Identity to Suit Your Needs - *Jeremy Sinclair*	**VSW10** (WPF + WinForms) * .NET Core = Modern Desktop - *Oren Novotny*	**VSW11** Feature Flags for Better DevOps - *Mickey Gousset*	**VSW12** Diving Deep into Entity Framework Core 2.x - *Philip Japikse*
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	**VSW13** Introduction to Webpack - *Chris Klug*	**VSW14** Make Your WPF Applications Shine with Fluent Design - *Jeremy Sinclair*	**VSW15** DevOps for Desktop Apps - *Oren Novotny*	**VSW16** Run Faster: Parallel Programming in C# - *Jeremy Clark*
7:30 PM	9:00 PM	Live! 360 Dessert Luau - Wantilan Pavilion			
Visual Studio Live! Day 3: Thursday, November 21, 2019					
7:30 AM	8:00 AM	Registration • Coffee and Morning Pastries			
8:00 AM	9:15 AM	**VSH01** Advanced Fiddler Techniques - *Robert Boedigheimer*	**VSH02** Release Management Strategies for Xamarin Developers - *Matthew Soucoup*	**VSH03** DevOps ICU: Improving Software Dev Results by (Correctly) Integrating UX - *Debbie Levitt*	**VSH04** Async internals in .NET - *Adam Furmanek*
9:30 AM	10:45 AM	**VSH05** Improving Web Performance - *Robert Boedigheimer*	**VSH06** Cross-Platform Development with Xamarin, Blazor, C#, and CSLA .NET - *Rockford Lhotka*	**VSH07** DevInSecOps - *Doyle M. Turner*	**VSH08** Internals of Exceptions - *Adam Furmanek*
11:00 AM	12:00 PM	**VISUAL STUDIO LIVE! PANEL DISCUSSION: Is There Such A Thing As A "Full Stack" Developer in 2019?** *Brian Randell (moderator), Heidi Araya, Adam Furmanek, Oren Novotny, Matthew Soucoup, & Alex Thissen*			
12:00 PM	1:00 PM	Lunch - *Oceana Ballroom*			
1:00 PM	2:15 PM	**VSH09** Overcoming Virtual Challenges: Strategies to Increase Engagement in Distributed Agile Teams - *Heidi Araya*	**VSH10** Place Your Bets: What Will Be the Next Breakthrough Dev Platform? - *Billy Hollis*	**VSH11** Growing your DevOps mindset - *René van Osnabrugge*	**VSH12** From .NET to .NET Core for Cloud Solutions - *Alex Thissen*
2:30 PM	3:45 PM	**VSH13** Closing The Feedback Loop: Unblock Your Agile/DevOps Transformation - *Heidi Araya & Esteban Garcia*	**VSH14** The Most Important Lessons Learned in Forty Years of Developing Software - *Billy Hollis*	**VSH15** (IAM) Secured - *Doyle M. Turner*	**VSH16** Building Real World Production—Ready Web API's with .NET Core - *Alex Thissen*
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, Thomas Sami Laiho, LaRock, Vishwas Lele, Rockford Lhotka, Karen Lopez, Matthew McDermott, Agnes Molnar, Brian Randell, Panu Saukko, Jen Stirrup, & Alex Thissen*			
Visual Studio 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	**VSF01** Workshop: Design an App UX in a Day - *Billy Hollis*		**VSF02** Workshop: Web Development in 2019 - *Chris Klug*	

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!

Accessing XML Documentation via Reflection

Zachary Patten

The .NET languages (C#, F# and Visual Basic) all support XML-formatted comments above types and members in source code. Aside from providing an easily intelligible standard for commenting code, these formatted comments are heavily integrated into Visual Studio and other development environments. They appear in tooltips and autocomplete suggestions, and in views like the Object Browser.

Even with all the benefits XML documentation currently provides, there's still a lot of untapped potential. You could use XML documentation to track bugs. You could integrate it with code analyzers to provide better recommendations. You could use it to control continuous integration pipelines. You could add documentation generation as a step in your automated pipeline so your public documentation is always up-to-date.

The main issue with using XML documentation for any of these purposes is that there are no methods in .NET to access it directly from code. However, with one loading function and a handful of extension methods, you can easily add the ability to access XML documentation via reflection.

This article discusses:

- Reflection
- XML documentation
- Extension methods
- Regular expressions
- XML file parsing

Technologies discussed:

Visual Studio, C#/F#/Visual Basic

Code download available at:

github.com/ZacharyPatten/Towel

The XML Documentation File

By default, the compiler won't do anything with XML documentation. You first have to enable the XML Documentation File option in Visual Studio, which is under the Build tab of a project's settings. Once enabled, the XML documentation will be extracted and placed into the designated file each time the code is built.

It's worth noting that the XML documentation settings are build-configuration-dependent. So you have to enable it on each build configuration you want it to run. Also, it's a good idea to make the name of the output XML file the same as the output assembly (just with an .xml file extension).

Figure 1 **The C# Source Code**

```
namespace Example
{
  /// <summary>XML Documentation on ExampleClass.</summary>
  public class ExampleClass
  {
    /// <summary>XML Documentation on ExampleMethod1.</summary>
    public static void ExampleMethod1() { }

    /// <summary>XML Documentation on ExampleNestedGenericClass.</summary>
    /// <typeparam name="A">Generic type A.</typeparam>
    /// <typeparam name="B">Generic type B.</typeparam>
    /// <typeparam name="C">Generic type C.</typeparam>
    public class ExampleNestedGenericClass<A, B, C>
    {
      /// <summary>XML Documentation on ExampleMethod2.</summary>
      /// <typeparam name="D">Generic type D.</typeparam>
      /// <typeparam name="E">Generic type E.</typeparam>
      /// <typeparam name="F">Generic type F.</typeparam>
      /// <param name="a">Parameter a.</param>
      /// <param name="d">Parameter d.</param>
      /// <param name="b">Parameter b.</param>
      /// <param name="e">Parameter e.</param>
      /// <param name="c">Parameter c.</param>
      /// <param name="f">Parameter f.</param>
      public static void ExampleMethod2<D, E, F>(
        A a, D d, B[] b, E[] e, C[,] c, F[,] f) { }
    }
  }
}
```

Figure 2 **The XML Output**

```
<?xml version="1.0"?>
<doc>
  <assembly>
    <name>Example</name>
  </assembly>
  <members>
    <member name="T:Example.ExampleClass">
      <summary>XML Documentation on ExampleClass.</summary>
    </member>
    <member name="M:Example.ExampleClass.ExampleMethod1">
      <summary>XML Documentation on ExampleMethod1.</summary>
    </member>
    <member name="T:Example.ExampleClass.ExampleNestedGenericClass`3">
      <summary>XML Documentation on ExampleNestedGenericClass.</summary>
      <typeparam name="A">Generic type A.</typeparam>
      <typeparam name="B">Generic type B.</typeparam>
      <typeparam name="C">Generic type C.</typeparam>
    </member>
    <member name="M:Example.ExampleClass.ExampleNestedGenericClass`3.
ExampleMethod2``3(`0,``0,`1[],``1[],`2[0:,0:,0:],``2[0:,0:,0:])">
      <summary>XML Documentation on ExampleMethod2.</summary>
      <typeparam name="D">Generic type D.</typeparam>
      <typeparam name="E">Generic type E.</typeparam>
      <typeparam name="F">Generic type F.</typeparam>
      <param name="a">Parameter a.</param>
      <param name="d">Parameter d.</param>
      <param name="b">Parameter b.</param>
      <param name="e">Parameter e.</param>
      <param name="c">Parameter c.</param>
      <param name="f">Parameter f.</param>
    </member>
  </members>
</doc>
```

Microsoft's language guides detail all the recommended tags, such as summary, param, typeparam, returns and remarks. However, you can create tags of your own for XML documentation. For example, I like to include my own tags like "citation" and "runtime." The compiler should extract any properly formatted XML tag from the comments.

The XML file produced at compile time is in a very simple format, with an assembly tag at the top to denote what assembly the file is documenting, and then every XML block from source code is placed into a separate member tag. The name property on the member XML tags represents the type/member of the code that the documentation is for. **Figure 1** includes C# source code and **Figure 2** shows the XML output from that source code.

The prefix of the name property determines what kind of code element the documentation is for, as follows: Methods "M:"; Types "T:"; Fields "F:"; Properties "P:"; Constructors "M:"; Events "E:".

The remaining parts of the name property are the fully qualified type and member names, along with any necessary parameters and/or generic parameters. Generic parameters from types are represented with a single apostrophe followed by the index "`X," while generic parameters from methods are represented with two apostrophes followed by the index "``Y." Arrays and unsafe pointers have the same syntax as they have in the source code; however, multidimensional arrays with a rank greater than one include "0:" strings separated by commas for each rank. Ref/Out/In parameters are all handled the same with just an at sign (@) appended to the end of the type. Optional parameters (with default values) don't have any special formatting.

As you can see, XML name properties can get a little complicated as you factor in various features of the languages, but the important takeaway is that the reflection types in .NET include all the logic necessary to build the "name" properties as they appear in the XML file.

Enough background—let's get into the code for accessing XML documentation. You first need to load the XML file into memory, which can be easily done using the XmlReader class. Content can be stored using Dictionary<string, string>. The key for the dictionary will be the name property as it exists in the XML file, and the value will be the content of the XML documentation (the inner XML of the member tag in the XML file). See **Figure 3**.

Next, let's explore accessing XML from the dictionary. The reflection types in the System.Reflection namespace represent types and members of compiled code: Type, FieldInfo, MethodInfo, ConstructorInfo, PropertyInfo, EventInfo, MemberInfo and ParameterInfo. You can create extension methods that let you call the methods as if they were instance methods on the reflection types. In the extension methods, you just need to convert the reflection type into the key of the dictionary that holds the loaded XML documentation. I created an extension method called GetDocumentation, as shown in **Figure 4**.

The extension methods for EventInfo and FieldInfo should be identical to the PropertyInfo method, just with "E:" and "F:" prefix strings, respectively. Note that the replacement of the "+" symbol with the "." is

Figure 3 **XML Loading Function**

```
internal static Dictionary<string, string> loadedXmlDocumentation =
  new Dictionary<string, string>();

public static void LoadXmlDocumentation(string xmlDocumentation)
{
  using (XmlReader xmlReader = XmlReader.Create(new StringReader(xmlDocumentation)))
  {
    while (xmlReader.Read())
    {
      if (xmlReader.NodeType == XmlNodeType.Element && xmlReader.Name == "member")
      {
        string raw_name = xmlReader["name"];
        loadedXmlDocumentation[raw_name] = xmlReader.ReadInnerXml();
      }
    }
  }
}
```

Figure 4 **Format the Key Strings**

```
// Helper method to format the key strings
private static string XmlDocumentationKeyHelper(
  string typeFullNameString,
  string memberNameString)
{
    string key = Regex.Replace(
      typeFullNameString, @"\[.*\]",
      string.Empty).Replace('+', '.');
    if (memberNameString != null)
    {
        key += "." + memberNameString;
    }
    return key;
}

public static string GetDocumentation(this Type type)
{
  string key = "T:" + XmlDocumentationKeyHelper(type.FullName, null);
  loadedXmlDocumentation.TryGetValue(key, out string documentation);
  return documentation;
}

public static string GetDocumentation(this PropertyInfo propertyInfo)
{
  string key = "P:" + XmlDocumentationKeyHelper(
    propertyInfo.DeclaringType.FullName, propertyInfo.Name);
  loadedXmlDocumentation.TryGetValue(key, out string documentation);
  return documentation;
}
```

to deal with nested types. The replacement of the bracketed string by "Regex" handles assembly information on the FullName of the Type.

The MethodInfo and ConstructorInfo extension methods are a little trickier. Constructors and methods can both have parameters—arrays, pointers, ref/in/out types, generic types and so forth. Methods can even define their own generic type parameters. It's easiest to start by storing all the generic parameters in dictionaries, like so:

```
Dictionary<string, int> typeGenericMap = new Dictionary<string, int>();
int tempTypeGeneric = 0;
Array.ForEach(methodInfo.DeclaringType.GetGenericArguments(),
  x => typeGenericMap[x.Name] = tempTypeGeneric++);

Dictionary<string, int> methodGenericMap = new Dictionary<string, int>();
int tempMethodGeneric = 0;
Array.ForEach(methodInfo.GetGenericArguments(),
  x => methodGenericMap.Add(x.Name, tempMethodGeneric++));

ParameterInfo[] parameterInfos = methodInfo.GetParameters();
```

With the generic parameters stored in dictionaries, you can easily obtain their indices. Remember that the generic parameters appear in the XML documentation as apostrophes followed by the index. However, generic parameters aren't the only special type you need to handle. Arrays, reference parameters, and pointers also have unique syntax in the XML file. **Figure 5** has some pseudo code for converting the ParameterInfos into strings.

Figure 5 is heavily simplified, but hopefully it gets the idea across. Check out my project on GitHub to see the full code for the ConstructorInfo and MethodInfo extension methods (github.com/ZacharyPatten/Towel).

MemberInfo is a base class for the reflection types. You can make an extension method for the MemberInfo type that funnels into the extension methods for the specific types, as shown in **Figure 6**.

Don't forget ParameterInfo. It has a Member property that returns the MemberInfo for the parameter. Just call the MemberInfo GetDocumentation extension method shown in **Figure 6** and extract the XML for the specific parameter if it exists. **Figure 7** shows how this is done.

Figure 5 **Parameters to Strings Pseudo Code**

```
foreach (var parameterInfo in parameterInfos) {
  if (parameterInfo.ParameterType.HasElementType) {
    // The type is either an array, pointer, or reference
    if (parameterInfo.ParameterType.IsArray) {
      // Append the "[]" array brackets onto the element type
    }
    else if (parameterInfo.ParameterType.IsPointer) {
      // Append the "*" pointer symbol to the element type
    }
    else if (parameterInfo.ParameterType.IsByRef) {
      // Append the "@" symbol to the element type
    }
  }
  else if (parameterInfo.ParameterType.IsGenericParameter) {
    // Look up the index of the generic from the
    // dictionaries in Figure 5, appending "`" if
    // the parameter is from a type or "``" if the
    // parameter is from a method
  }
  else {
    // Nothing fancy, just convert the type to a string
  }
}
```

Figure 6 **MemberInfo Extension Method**

```
public static string GetDocumentation(this MemberInfo memberInfo)
{
  if (memberInfo.MemberType.HasFlag(MemberTypes.Field)) {
    return ((FieldInfo)memberInfo).GetDocumentation();
  }
  else if (memberInfo.MemberType.HasFlag(MemberTypes.Property)) {
    return ((PropertyInfo)memberInfo).GetDocumentation();
  }
  else if (memberInfo.MemberType.HasFlag(MemberTypes.Event)) {
    return ((EventInfo)memberInfo).GetDocumentation();
  }
  else if (memberInfo.MemberType.HasFlag(MemberTypes.Constructor)) {
    return ((ConstructorInfo)memberInfo).GetDocumentation();
  }
  else if (memberInfo.MemberType.HasFlag(MemberTypes.Method)) {
    return ((MethodInfo)memberInfo).GetDocumentation();
  }
  else if (memberInfo.MemberType.HasFlag(MemberTypes.TypeInfo) ||
    memberInfo.MemberType.HasFlag(MemberTypes.NestedType)) {
    return ((TypeInfo)memberInfo).GetDocumentation();
  }
  else {
    return null;
  }
}
```

Figure 7 **ParameterInfo Extension Method**

```
public static string GetDocumentation(this ParameterInfo parameterInfo)
{
  string memberDocumentation = parameterInfo.Member.GetDocumentation();
  if (memberDocumentation != null) {
    string regexPattern =
      Regex.Escape(@"<param name=" + "\"" + parameterInfo.Name + "\"" + @">") +
      ".*?" +
      Regex.Escape(@"</param>");
    Match match = Regex.Match(memberDocumentation, regexPattern);
    if (match.Success) {
      return match.Value;
    }
  }
  return null;
}
```

Automatically Loading the XML Files as Needed

Is there a way to bypass the loading function? If you follow the standard of having your output XML files in the same directory and with the same name as your assemblies, it's easy to automatically look up an XML file so you don't need to call the loading function.

You can get the assembly from any type with the GetAssembly method or the Assembly property. Then you can get the file location of an assembly through the CodeBase property of the assembly. Finally, just alter the file path to look for the XML file instead of the assembly, and call the loading function shown in **Figure 4**. You can see how this works in **Figure 8**.

Then you can update the extension methods to load the XML documentation of the assembly if it hasn't already been loaded. The following code shows the System.Type extension method from before with automatic XML file loading:

```
public static string GetDocumentation(this Type type)
{
  LoadXmlDocumentation(type.Assembly);

  // ... Rest of the code
}
```

Usage Examples

Now that you have all of the necessary framework code, you just need to call it. Don't forget to add a using statement at the top of the file so the extension methods are visible. **Figure 9** shows usage examples for printing the XML documentation of the current assembly to the console.

Figure 8 **Loading XML Documentation from Assembly**

```
public static string GetDirectoryPath(this Assembly assembly)
{
  string codeBase = assembly.CodeBase;
  UriBuilder uri = new UriBuilder(codeBase);
  string path = Uri.UnescapeDataString(uri.Path);
  return Path.GetDirectoryName(path);
}

internal static HashSet<Assembly> loadedAssemblies = new HashSet<Assembly>();

internal static void LoadXmlDocumentation(Assembly assembly)
{
  if (loadedAssemblies.Contains(assembly)) {
    return; // Already loaded
  }
  string directoryPath = assembly.GetDirectoryPath();
  string xmlFilePath = Path.Combine(directoryPath, assembly.GetName().Name + ".xml");
  if (File.Exists(xmlFilePath)) {
    LoadXmlDocumentation(File.ReadAllText(xmlFilePath));
    loadedAssemblies.Add(assembly);
  }
}
```

The most obvious use case for this code is documentation generation. You could easily reflect through all the types and members in your assembly, grab the XML documentation from them, and dump that documentation into any format you want.

There are multiple documentation generators out there, but they're often rather restrictive as to their output formats. What I wanted was to dump all the documentation into a simple HTML tree that I could style myself. So, writing my own documentation generator using this methodology was the easiest solution. Also, none of the documentation generators are able to handle my custom XML tags. Here are examples of custom XML tags to give you some tag ideas:

- notes: adding additional documentation outside the standard tags
- revision: help document the history of a member
- todo: mark members for future development
- link: URL link to further documentation and/or videos
- critical: mark members that are critical for external projects and shouldn't be edited
- bug: express a known bug in a member until it can be fixed
- test: link to the unit tests for a member, or perhaps metadata for the unit testing
- runtime: big O-notation runtime complexity
- citation: crediting sources for code

If you write a documentation Web site generator that makes use of these kinds of tags and you use continuous integration to deploy the Web site on code commits, then notifying users of bugs could be as simple as adding the "bug" XML tag to one of your members in source code. Nice.

You could also use XML documentation for code analysis. If you use standardized XML tags like "runtime," it may help you resolve hot spots in code. If one method is documented to have a runtime complexity of "O(n)," while another overload of the method has a runtime complexity of "O(ln(n))," you'd clearly want to use the latter if possible. Theoretically, this could be added to code analyzers or possibly Visual Studio extensions to offer coding suggestions.

There are already compilation warnings to encourage XML documentation on publicly visible types and members in .NET code, but there aren't any warnings to encourage people to use the "exception" tag. Being able to grab the XML through reflection may allow you to more easily find members that don't meet development and documentation standards.

Figure 9 **Usage Examples**

```
// Optional loading function
LoadXmlDocumentation(File.ReadAllText("PATH/TO/XML/FILE.xml"));

// Write the documentation of all types to the console
foreach (Type type in Assembly.GetExecutingAssembly().GetTypes()) {
  Console.WriteLine(type.GetDocumentation());
}

// Write all the documentation of every member to the console
foreach (Type type in Assembly.GetExecutingAssembly().GetTypes()) {
  foreach (MemberInfo memberInfo in type.GetMembers()) {
    Console.WriteLine(memberInfo.GetDocumentation());
  }
}

// Write all the documentation of every parameter to the console
foreach (Type type in Assembly.GetExecutingAssembly().GetTypes()) {
  foreach (MemberInfo memberInfo in type.GetMembers()) {
    if (memberInfo is MethodBase) {
      MethodBase methodBase = memberInfo as MethodBase;
      foreach (ParameterInfo parameterInfo in methodBase.GetParameters()) {
        Console.WriteLine(parameterInfo.GetDocumentation());
      }
    }
  }
}
```

Important Mentions

This solution is not future proof. As new features are added in future versions of languages like C# and Visual Basic, the format of the XML file may change. It's unlikely that backward compatibility would be broken, but new features with new XML formats would require the extension methods to be modified to deal with them.

XML documentation and attributes are very similar. Both appear above members in code providing metadata about the member. However, XML documentation shouldn't be considered an alternative to attributes. Attributes are guaranteed to be reachable at runtime, because they're included in the compiled code. Any metadata necessary for the functionality of code should be implemented as an attribute.

I encourage you to look at the code in my Towel project on GitHub (github.com/ZacharyPatten/Towel). There are also unit tests for the code within the project. I'm sure there's plenty of room for improvement in the code, and probably some test cases I missed when testing it. You could go a step further and write extension method overloads that take string parameters to extract specific tags from the XML, such as grabbing the "summary" content specifically.

I hope it's now clear that you can indeed access XML documentation easily via reflection. Probably the biggest drawback is having to make sure that the XML exists in the necessary file location so it can be loaded into memory. Just be sure to check that the files exist, and this technique should do the trick until a better methodology comes along. ■

Zachary Patten *is just another programmer who likes coffee, Power Rangers and dogs. He graduated from Kansas State University, and currently has approximately five years of professional programming experience.*

Thanks to the following Microsoft technical experts for reviewing this article: *Den Delimaschi, Bill Wagner*

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: To Be Announced** - *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!

Exploring Blockchain Consensus

Erik Zhang and John deVadoss

Blockchain platforms have led to incredible advances in designing and developing decentralized applications and systems, and have been applied toward domains ranging from cryptocurrencies to enterprise supply chains. While the applications are vast, they're all based on a core set of design patterns that advance the state of the art in the theory and practice of distributed systems.

A blockchain is a monotonically increasing list of records (or blocks) that are linked together using cryptographic techniques. Blocks consist of valid transactions that are hashed and encoded into a Merkle tree, and each block contains a cryptographic hash of the previous block in the chain. This ensures the integrity of the blockchain and enables the relatively inexpensive verification and the independent audit of the transactions in each block and across the chain. A blockchain intrinsically is public and tamper-proof, meaning existing blocks cannot be altered in any manner.

Core to the blockchain is the model of the *ledger*, an unalterable, append-only log of the transactions that take place across various entities. To maintain the integrity of the ledger, the various entities need a way to "agree" or to reach consensus on which set of incremental transactions (or blocks) are to be appended to the ledger.

The consensus problem is a well-known and fundamental computer science problem in the coordination and control of multi-entity systems. A simplistic approach is of course for all the entities to agree on a majority value. However, one or more faulty entities can skew the outcome, resulting in consensus that is unachievable or incorrect.

In this article, we explore the topic of consensus for blockchains, and share a practical, real-world implementation built on the .NET Core platform using C# that's used by the NEO blockchain, the Binance Exchange and other organizations.

Let's start by looking at blockchain platforms, which are programmable blockchains that enable developers to envision and build truly decentralized applications. These can span all manner of markets, including financial markets, gaming, enterprise consortiums, sports, health care networks, sovereign identities, real estate and other asset markets and more. Blockchain platforms such as Ethereum and NEO serve as decentralized application platforms that provide the foundation for a new application model for developers.

At their core, blockchain platforms are distributed systems, building on a foundation of theory and practice that spans decades of computer science research. While there are many recurring patterns and principles, Blockchain platforms have revolutionized

This article discusses:

- Distributed systems and the state machine approach
- Byzantine Fault Tolerance and the dBFT algorithm
- Building a blockchain implementation with .NET Core and C#

Technologies discussed:

Blockchain, Blockchain Platforms, C#, .NET Core

the theory of distributed systems in how we deal with trust. In the next section, we drill down further into distributed systems and their implementation using the well-known state machine model in computer science.

Distributed Systems and the State Machine Approach

Distributed systems share a core set of special characteristics, as explored in the field of theoretical computer science. These include:

Concurrency: Multiple activities across the distributed system may be executed simultaneously and independently of each other. This implies that there's a need for coordination across the different flows of execution.

Independent failure modes: Multiple components across the distributed system may fail independently of each other.

No global time: Multiple flows of execution may be aligned with spatially independent local clocks. Even in the event that these clocks are initially synchronized, clock drift will eventually result. This implies that time and event ordering is a core challenge in distributed systems.

Communications delay: There's an inherent lag in how events and their side effects propagate through the distributed system.

Inconsistent state: Concurrency, independent failure modes, and communications delays together imply that the view of any state will not be consistent throughout the distributed system.

Collectively, these characteristics require that distributed systems be designed to be fault-tolerant, in order to continue to operate in the event of one or more faults (or complete failure) of one or more subsystems.

> There's an inherent lag in how events and their side effects propagate through the distributed system.

The state machine approach is a general method for implementing a fault-tolerant distributed system by replicating services and coordinating client interactions across service replicas. A replicated state machine is deterministic in that it consists of a set of state variables that encode its state and transactions. These state variables can cause the machine to transition from one valid state to the valid next state. Each transaction is executed deterministically (that is, transactions are atomic). Essentially, a replicated state machine is a distributed set of services where all the services start with the same initial state and then agree (that is, reach consensus) on each of the subsequent state transitions.

Consensus Across Replicated State Machines

Formally, the goal of a consensus algorithm is to satisfy three key properties. These are:

Termination: All non-faulty services in the system eventually decide on some output value. This is often referred to as *liveness*.

Integrity: If all of the non-faulty services propose the same output value, then any non-faulty service must decide on the same output value. A weaker form of integrity is one where the output value must equal a value that was proposed by some non-faulty service (not necessarily all of them).

Agreement: All non-faulty services in the system eventually decide on the same output value. This is often referred to as *safety*.

Distributed systems theory has made tremendous leaps in the understanding of consensus algorithms, but consensus in a completely asynchronous distributed system has proven impossible to achieve in the presence of even a single faulty service. This is called the FLP impossibility, named after the researchers (Michael J. Fischer, Nancy Lynch and Mike Patterson) who posited a definitive upper bound on what's possible to achieve with distributed processes in an asynchronous environment.

The FLP impossibility has spawned research spanning two innovative approaches. One set of algorithms relies on the so-called Nakamoto consensus. It applies an unconventional approach that relies on non-determinism to address the inherent scale challenges in attempting to generate consensus in a distributed system. The brilliance of the Nakamoto consensus is that rather than every service agreeing on a value, the algorithm focuses on all of the services agreeing on the *probability* of the value being correct. However, this results in probabilistic agreement—that is, the lack of deterministically finalizing a value at every state transition creates a situation where there's no guarantee of true finality. This leads to the so-called forking scenario with respect to the distributed system. For this reason, we'll ignore the Nakamoto consensus for the remainder of this article.

A second set of practical fault-tolerant consensus algorithms has assumed some level of synchrony assumptions in order to make progress. What this means is that some protocols are designed to work in unreliable networks—such as, say, the Internet—that drop messages and may cause arbitrary delay, while other protocols are optimized for highly reliable network channels. These protocols are said to operate under different sets of synchrony assumptions. Synchrony assumptions may be explicit or implicit by relying on leader election algorithms, for instance. Consensus algorithms that are based on leader election are called Paxos algorithms.

Byzantine Fault Tolerant Consensus

Byzantine failures pose a challenge for leader-based consensus algorithms. These failures occur when components or sub-components of a distributed system fail, and there's imperfect information about whether a component (or sub-component) has actually failed. Algorithmic proofs exist to demonstrate that a malicious leader can't cause inconsistency, but distributed systems theory has yet to demonstrate that a malicious leader can't prevent progress.

The so-called practical BFT (pBFT) algorithm by Castro and Liskov was the first attempt to describe an algorithm by which the system can detect lack of progress and choose a new leader. pBFT was devised to address the twin flaws in previous attempts—either the algorithm was too slow to be of practical use or synchrony had to be assumed to satisfy the "agreement" property.

The pBFT algorithm demonstrated that it could provide both liveness and safety as long as a maximum of (n - 1) / 3 services were faulty in the distributed system. pBFT cycles through a succession of "views" with each view having one primary service acting as the leader and the remaining services acting as backups. At a conceptual level, the pBFT algorithm works as follows:

1. The client sends a request to the primary (leader) service.
2. The primary (leader) service broadcasts the request to all of the backup services.
3. The primary and the backup services perform the work requested and then send back a reply to the client.
4. The request is served successfully when the client receives m+1 responses from the different services spanning the distributed system with the same result, where m is the maximum number of faulty services allowed.

The primary (leader) service is changed during every view (round of consensus) and may be substituted if a predefined quantity of time has passed without the leader broadcasting a request to the backups. As long as the leader is non-faulty, pBFT works reasonably well; however, the process of replacing a faulty leader is highly inefficient.

pBFT improved on the existing theory, but in practice it's not suitable for real-world scenarios due to its inherent scalability challenges and its inability to distinguish malicious behavior from transient communication faults.

Delegated Byzantine Fault Tolerance

Enter Delegated Byzantine Fault Tolerance (dBFT), which was proposed by Erik Zhang, founder of the NEO blockchain in 2014. dBFT extends pBFT concepts to the state machine replication scenario, and provides the first practical, public access to fast single-block data finality (about 15 seconds). dBFT is now in use by the NEO blockchain, the Binance Exchange and other major platforms globally.

The key innovation in Zhang's proposal was to distinguish consensus nodes (services that can participate in the consensus algorithm to propose new state changes and vote) and ordinary nodes (services that can execute the atomic transactions and transition state, but don't take part in the consensus algorithm and can't propose new state changes). In doing so, dBFT became the first practical BFT to function at scale, addressing the challenges inherent in pBFT.

The C# implementation of dBFT is available in the public domain (MIT License) on GitHub at bit.ly/2Zl1Sem.

> Erik Zhang, the founder of the NEO blockchain, proposed the Delegated Byzantine Fault Tolerance algorithm (dBFT) in 2014, extending pBFT concepts to the state machine replication scenario.

The dBFT algorithm comprises three distinct phases, Pre-Prepare, Prepare and Persist. Before we explore each of these phases, let's take a moment to clarify terminology and the algorithmic steps involved.

N: The number of active consensus nodes

f: The number of Byzantine (that is, malicious) nodes, with f being no more than (N - 1) / 3

v: The current view number (each view is a new round or attempt at consensus)

b: The proposed block of atomic transactions, the execution of which transitions the system to the next valid state

p: The index of the speaker, that is the leader for this view which proposes the block. The speaker and the remaining delegates together comprise the **N** consensus nodes.

At a conceptual level, dBFT comprises the following steps:

1. A cryptographically signed transaction is "broadcast" by a client to the nodes in the distributed system.
2. The **N** consensus nodes receive the transaction and collect them into their in-memory pool of transactions.
3. For the current view, the unique speaker **p** packages the trans-

Figure 1 **The MakePrepareRequest Method**

```
public ConsensusPayload MakePrepareRequest()
{
  byte[] buffer = new byte[sizeof(ulong)];
  random.NextBytes(buffer);
  List<Transaction> transactions =
    Blockchain.Singleton.MemPool.GetSortedVerifiedTransactions()
    .Take((int)NativeContract.Policy.GetMaxTransactionsPerBlock(Snapshot))
    .ToList();
  TransactionHashes = transactions.Select(p => p.Hash).ToArray();
  Transactions = transactions.ToDictionary(p => p.Hash);
  Block.Timestamp = Math.Max(TimeProvider.Current.UtcNow.ToTimestampMS(),
    PrevHeader.Timestamp + 1);
  Block.ConsensusData.Nonce = BitConverter.ToUInt64(buffer, 0);
  return PreparationPayloads[MyIndex] = MakeSignedPayload(new PrepareRequest
  {
    Timestamp = Block.Timestamp,
    Nonce = Block.ConsensusData.Nonce,
    TransactionHashes = TransactionHashes
  });
}
```

Figure 2 **The SendPrepareRequest Method**

```
private void SendPrepareRequest()
{
  Log($"send prepare request: height={context.Block.Index} view={context.
ViewNumber}");
  localNode.Tell(new LocalNode.SendDirectly { Inventory = context.
MakePrepareRequest() });

  if (context.Validators.Length == 1)
    CheckPreparations();

  if (context.TransactionHashes.Length > 0)
  {
    foreach (InvPayload payload in InvPayload.CreateGroup(InventoryType.TX,
      context.TransactionHashes))
      localNode.Tell(Message.Create(MessageCommand.Inv, payload));
  }
  ChangeTimer(TimeSpan.FromMilliseconds((Blockchain.MillisecondsPerBlock <<
    (context.ViewNumber + 1)) - (context.ViewNumber == 0 ?
    Blockchain.MillisecondsPerBlock : 0)));
}
```

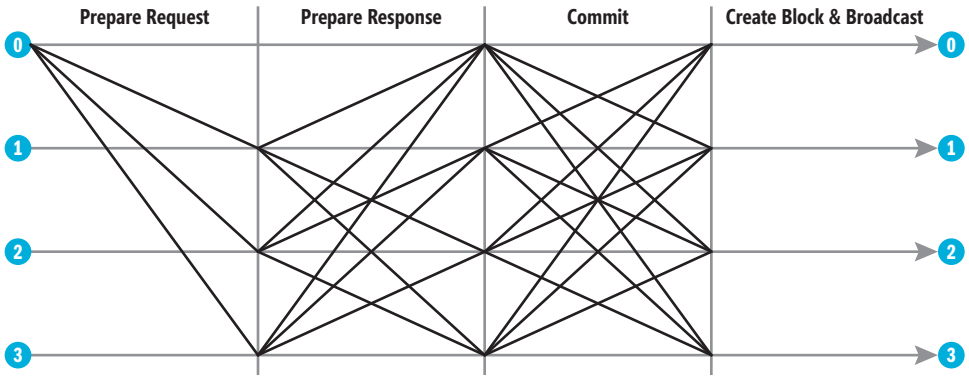


Figure 3 **The dBFT Consensus Algorithm Phases**

actions from the memory pool into a new proposal for block **b**. **Figure 1** illustrates the MakePrepareRequest method and **Figure 2** illustrates the SendPrepareRequest method.

4. The N-1 remaining delegates receive the new proposed block **b**, verify it and then broadcast the verification. For brevity, the code snippets for the remaining steps aren't included inline and are on GitHub at the link provided earlier.
5. Any consensus node, on receiving at least (**N** - **f**) verifications reaches consensus and then publishes the new block.
6. Any node, on receiving a new block, deletes all the transactions from the in-memory pool, and if it's a consensus node starts on the next view (round of consensus).

With all that settled, let's return to the three phases of the dBFT algorithm. They are:

Pre-Prepare: The speaker for the view is responsible for broadcasting the Prepare-Request message to the delegates and for initiating the new proposed block of transactions.

Prepare: On receiving the Pre-Prepare message, the delegates broadcast the Prepare-Response message if the proposed block is successfully verified. On receiving (**N** - **f**) successful block verifications, the consensus nodes enter the next phase.

Persist: The nodes publish a new block and enter the next round of consensus.

The first version of dBFT was susceptible to a single block fork due to network latency in certain edge cases.

In the event that consensus isn't reached in a certain round (view), the consensus nodes may initiate a proposal to change the view, with a new speaker (leader) and restart the activity of reaching consensus, after receiving at least (**N** - **f**) proposals to change the view with the exact same view number. The wait time to propose a new round increases exponentially in order to avoid frequent view changes and to ensure consensus within practical time bounds.

The first version of dBFT was susceptible to a single block fork due to network latency in certain edge cases. Essentially, the fact that nodes could time-out after sending a PrepareResponse message meant that nodes could time-out and transition at slightly different times. In the event that only one consensus node didn't time-out and that node had already received 2**f** PrepareResponse messages, it would then generate a valid block while the other consensus nodes would've moved on to the next view. There, those consensus nodes could in theory achieve consensus and sign another block of transactions at the same level. While this scenario could transpire without blocking consensus, one or more nodes could accept the forked block and stall.

dBFT 2.0 addressed this by adding a new commit phase. To prevent potential stalling, dBFT 2.0 also augments the consensus algorithm with a recovery message implementation. This recovery mechanism has the additional benefit of significantly improving block times in cases where network latency is degraded due to the network being compromised.

Wrapping Up

Distributed systems have been fundamental in transforming the computing industry, and by extension how we conduct commerce globally and how we universally engage as a community. The emergence of blockchains has spurred developers to study and to scrutinize well-established principles and paradigms in distributed systems, and in doing so has catalyzed a wave of innovation that continues to break new ground in how developers build the next generation of software applications.

In this article we've focused on the topic of consensus and illustrated a pioneering new approach with the dBFT consensus algorithm. While we used C# to illustrate the dBFT algorithm for consensus, we've implemented the dBFT algorithm in C++, Python, Typescript and Go, to name a few languages. We hope this article provides developers with a better understanding of blockchain consensus and enables them to use and to build on top of the pioneering dBFT algorithm. ■

Erik Zhang *is founder and core developer of NEO, author of the dBFT consensus mechanism, expert on blockchain technology and computer security, and a certified information system auditor (CISA). Previously he was engaged in Shanda Games and huobi.com, where he specialized in information security and R&D in digital currency.*

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 startups. deVadoss did his Ph.D. work in machine learning, specializing in recurrent neural networks.*

Thanks to the following technical experts for reviewing this article: *Chuan Lu, Harry Pierson*

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**

SAVE $300 **With Early Bird Savings. Register by October 25!**
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: To Be Announced**		
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: To Be Announced**		
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!

DINO ESPOSITO

Using gRPC in a Microservice Architecture

The microservice architecture style has been taking the software industry by storm in the recent past and with some good arguments. It objectively delivers key benefits that make the resulting system resilient to failure and scalable at the desired level of granularity. In an enterprise scenario, a microservice architecture enables the use of heterogeneous technologies and different architectural patterns. It also dramatically simplifies deployment and allows the engineering department to more easily shape up a bespoke project that reflects the internal organization.

All that said, what's the concrete result of designing and deploying a microservice-based architecture? The commonly accepted answer to this question is the trigger for this month's article, and the logical background for introducing the gRPC framework in a microservice architecture.

Breaking Up the Monolith

Although no official definition of microservices probably exists, nearly everybody agrees on a definition that goes along the lines of organizing the entire logic of a single software application in a collection of smaller services. Each constituent service ends up being considerably smaller in size and complexity than the single application in its entirety. These (micro) services are treated as independent units of the application and are implemented and deployed autonomously. Also important, is each microservice be fully responsible for its own data.

In a nutshell, the microservice architecture breaks up a single software monolith into a myriad of independent and distributed components. Note that in this context the term myriad is a bit of a hyperbole, with the number of microservices usually measured on the order of tens. A smart (if relative and non-quantitative) definition of the ideal microservice size is that it should never be larger than a problem's subdomain and never smaller than a cluster of related domain objects.

In a microservice application, the individual components form a distributed architecture in which each component runs its own process and communicates with others over possibly lightweight protocols. There are many ways in which two distinct microservices can communicate over the network. Most of the time, they do it using REST interfaces and JSON for data serialization. The whole Web services world was devised to use SOAP as the underlying protocol for communication and XML as the data serialization format. SOAP wasn't necessarily ideal, but it was the best the industry could work out 20 years ago. Later came REST and RESTful interfaces, which offered important advantages over SOAP with its well-defined and standardized messaging patterns (like the WS-XXX family).

REST arrived as an antidote to the perceived complexity of SOAP, performing direct access to the data and defined in the form of resources. Access to resources occurs using the verbs (and the rules) of the underlying HTTP protocol. In other words, a RESTful interface has no need to define interfaces and constraints, other than those set by the underlying HTTP protocol itself. In this picture, JSON is only a serialization format, less verbose and more flexible than XML. Today REST and JSON together represent the lion's share of microservices communications, but there are other, and possibly better, communication technologies to consider. The most enticing is the gRPC framework.

Bringing gRPC to the Table of Microservices

At first, using the gRPC framework to connect distributed components may look like a step backward. After celebrating the advent of REST as the triumph of flexibility and decoupling, the industry is now carefully considering an approach centered on the concept of remote procedure calls (RPC). By design, an RPC requires some level of coupling between the caller and receiver.

In addition, the gRPC framework pushes well-defined messages, versioning and contracted endpoints. The framework runs over the newer HTTP/2 protocol and exchanges data in a binary format. Under the hood, a number of other, subtler differences exist that make gRPC worth a closer look. But is gRPC a better REST, or is it just particularly suited to a microservices scenario?

The point is that REST in itself is only the formalization of an ideal uniform interface to connect components. Inspired in 2000 by the way the Web was developing, REST never really achieved the goal of enabling any RESTful client to talk to any RESTful server regardless of knowing much of the details. How many RESTful Web APIs return 200 OK to report an error? And how many use the same POST verb to add, delete and update resources? REST definitely works, but beyond hype and fanfare it's nothing more than a plain API exposed over HTTP that just requires an HTTP call to return a response. Each developer calling any endpoint of any declared REST API needs to know all of its details and, often, he is free to place calls in a different way than is documented and still get a valid response. As brutal as it may sound, there's no (concretely applied) standard around REST.

As implemented in most real-world applications today, REST is basically a simplified, unstructured and improvable form of RPC in which HTTP elements (headers, query string, payloads, status

codes) form the syntax of each call, and every call in every service may feature different characteristics.

So where does gRPC come from? It was originally developed by Google as a framework to better connect its large internal network of standalone services. That framework was open sourced in 2015 under the name gRPC, and is now owned by the Cloud Native Computing Foundation. The gRPC framework addressed precisely the scenario of most of today's microservice-style architectures.

In which way is gRPC preferable over a REST-inspired HTTP API? Let's take a look.

Contrasting gRPC and HTTP API

At its core, the gRPC framework is designed for any scenario where communication efficiency is critical. This marks a first crucial difference from the HTTP API. Two microservices are not like two communicating Web sites. For example, the ASP.NET application that consumes the API exposed by another Web site evolves independently from the site. While there's still no realistic guarantee that the HTTP API will evolve, preserving backward compatibility toward all of its callers, the link between API owner and callers remains quite loose.

The gRPC framework instead enforces a binding between client and service. In this regard, it represents a viable option in circumstances where client and service are managed by the same team and follow the same development cycle. In other words, tight coupling is ideal in a tightly coupled network of components like a microservice-oriented application. Once it's determined that the tight-coupling argument isn't an architectural shortcoming, then the benefits of bindings between client and service can be further analyzed.

The major benefit of gRPC over a plain HTTP API is performance. As mentioned, the gRPC framework uses HTTP/2 as the transportation layer, which is the newest version of HTTP with some specific improvements in the area of binary framing, compression and multiplexing over a single connection. As a result, HTTP/2 is more efficient and also flexible both in sending and receiving packets. Beyond the transportation protocol, the framework itself takes care of serializing and deserializing data using the Protobuf binary format. This contributes to making the traffic generated by each gRPC call more compact.

Beyond performance, the gRPC framework provides benefits also in the application development space. Each service is based on a .proto file that acts as the contract of the service. The contract can be used to generate the service and client. The framework takes care of the dirty job of generating client, service base class and messages for every major computer language, with all of them using the same contract standard. Messages in particular are crucial as any method is designed to receive a message and return a message, and the contract of both is fixed. This means that the call fails if anything is sent or received outside the boundaries of the contract. Anybody who ever experienced the burden of receiving incorrect or inconsistent JSON or weird status codes knows very well what I mean here. The gRPC framework is type-safe and consistent across all supported platforms and implementations.

In a nutshell, the gRPC framework was somehow to avoid the shortcomings of a RESTful, HTTP-based API in the context of a (very) large organization and myriads of distributed components.

Microservice-to-Microservice Communication

In past installments of this column, I covered the basics of gRPC and discussed some examples of its capabilities. In particular, last month I wrote about the streaming API. Leveraging the low-level capabilities of HTTP/2, the gRPC framework allows the client to send, and the service to return, multiple packets per connection. The interesting thing is that both the client and the service can be sending and returning multiple packets per connection, thus turning a gRPC communication into a sort of superfast chat.

One of the most talked about anti-patterns of SOAP and REST, and distributed architecture in general, was the "chatty" style of an API in which the client needs to place multiple calls in order to receive all the data it logically needs. While the "chatty" style remains an anti-pattern to avoid, the advanced streaming capabilities of the framework make it possible to arrange a bi-directional channel between two endpoints so messages can be exchanged in real time without polling.

Think, for example, of a monitoring scenario in which the service receives data constantly from a connected device, does some work on it and then passes data to a real-time Web dashboard. In an architecture in which the aggregator and the real-time data provider microservice communicate directly, the use of the gRPC protocol for bi-directional communication can realistically increase the performance of every single call by at least a few milliseconds.

Doesn't it sound like a huge gain? Well, that's probably why gRPC isn't (or not yet) a good fit for the Web UI. But for back-end microservice-to-microservice communication, especially in applications with heavy loads where every bit counts, it could really make a difference.

When Using gRPC between microservices the front end is connected to a microservice (that is, the gateway to the microservice cluster) via a plain HTTP API consumed over the Web. As mentioned, at the current stage of both gRPC and the browser technology, no gRPC services can be invoked directly from any browser. Most modern browsers support HTTP/2, but only in a sort of transparent way and only after negotiating the protocol to use during the initial connection to the server. Because of this, browsers don't allow clients to dictate the protocol to use and therefore can't directly invoke the gRPC service. Microservice-to-microservice communication or, more broadly, server-to-server communication, is a good scenario for gRPC, but mobile and desktop applications can use gRPC, as well.

Wrapping Up

Until recently, the word REST has been used to label nearly any public endpoints exposed over the Web through the HTTP protocol. REST came with the promise that once resources are defined the entire public API results from the combination of resource names and HTTP verbs. As more verbs and resources are added, new and existing clients can just connect and place calls. This flexibility comes at an obvious cost—the lack of strict rules and contracts. This is just what gRPC brings to the table. ■

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.*

AGENDA-AT-A-GLANCE

#LIVE360

AI Application Development	Analytics	Bots	Data Science and Machine Learning	Internet of Things

START TIME	END TIME		
		Artificial Intelligence 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	**AIS01** Hands-On Lab: Build Your Own A.I. Powered Robot - *Henk Boelman*	
2:00 PM	7:00 PM	Pre-Conference Registration - Royal Pacific Resort Conference Center	
START TIME	**END TIME**	**Artificial Intelligence Live! Pre-Conference Workshop: Monday, November 18, 2019**	
7:00 AM	8:30 AM	Registration • Coffee and Morning Pastries	
8:30 AM	5:30 PM	**AIM01** Workshop: Cloud Scale Machine Learning (Hands-On) - *Seth Juarez & Cassie Breviu*	**AIM02** Workshop: Big Data and Machine Learning with Hadoop, Spark, and SQL Server 2019 - *Andrew Brust*
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**	**Artificial Intelligence Live! Day 1: Tuesday, November 19, 2019**	
7:00 AM	8:00 AM	Registration • Coffee and Morning Pastries	
8:00 AM	9:00 AM	**ARTIFICIAL INTELLIGENCE LIVE! KEYNOTE: From Zero to AI Hero – Automatically Generate ML Models Using Azure Machine Learning Service, Automated ML** - *Sujatha Sagiraju, Group Program Manager, Azure Cloud & AI Group, Microsoft*	
9:15 AM	10:30 AM	**AIT01** Make Your App See, Hear and Think with Cognitive Services - *Roy Cornelissen*	**AIT02** Data Insights with End-2-End Azure Analytics - Scott Klein
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	**AIT03** Fast Focus: Personal Assistants vs. Bots - *Brian Randell*	**AIT04** Fast Focus: The Sunny Side of AI - *Henk Boelman*
2:00 PM	2:20 PM	**AIT05** Fast Focus: AR vs. VR vs. MR - What Does it All Mean? - *Nick Landry*	**AIT06** Fast Focus: AutoML Crash Course - *Andrew Brust*
2:20 PM	2:45 PM	Networking Break • Visit the EXPO - *Pacifica 7*	
2:45 PM	4:00 PM	**AIT07** Did You Know Cognitive Services Could Do This? A Computer Vision Quest - *Andreas Erben*	**AIT08** AI and Analytics with Apache Spark on Azure Databricks - *Andrew Brust*
4:15 PM	5:30 PM	**AIT09** Bot Building 101 with the Microsoft Bot Framework - *Brian Randell*	**AIT10** Diving Into Deep Learning with Databricks - *Ginger Grant*
5:30 PM	7:30 PM	Exhibitor Reception - *Pacifica 7*	
START TIME	**END TIME**	**Artificial Intelligence Live! Day 2: Wednesday, November 20, 2019**	
7:30 AM	8:00 AM	Registration • Coffee and Morning Pastries	
8:00 AM	9:15 AM	**AIW01** An Introduction to HoloLens & Mixed Reality for the Enterprise - *Nick Landry*	**AIW02** Machine Learning 101 for Developers - *Jen Underwood*
9:30 AM	10:45 AM	**AIW03** Building a Holographic AI Assistant with ASP.NET Bots, Natural Language, & Mixed Reality - *Nick Landry*	**AIW04** Getting Started with Azure Machine Learning Services - *Henk Boelman*
10:45 AM	11:30 AM	Networking Break • Visit the EXPO - *Pacifica 7*	
11:30 AM	12:30 PM	**LIVE! 360 KEYNOTE: To Be Announced**	
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	**AIW05** Practical Internet of Things for the Microsoft Developer - *Eric D. Boyd*	**AIW06** DevOps for Artificial Intelligence, the Road to Production - *Henk Boelman*
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	**AIW07** The Edge of Tomorrow. Use Visual Studio Code and Raspberry Pi to Create IoT Edge Solutions - *Nicholas McCollum*	**AIW08** Visualization Best Practices for Machine Learning Applications - *Jen Underwood*
7:30 PM	9:00 PM	Live! 360 Dessert Luau - *Wantilan Pavilion*	
START TIME	**END TIME**	**Artificial Intelligence Live! Day 3: Thursday, November 21, 2019**	
7:30 AM	8:00 AM	Registration • Coffee and Morning Pastries	
8:00 AM	9:15 AM	**AIH01** Machine Learning the .NET Way - *Bri Achtman*	**AIH02** Machine Learning with Power BI - *Raj Krishnan*
9:30 AM	10:45 AM	**AIH03** Demystifying User Management for Voice Apps - *Heather Downing*	**AIH04** Launching A Data Science Project: Cleaning Is Half the Battle - *Kevin Feasel*
11:00 AM	12:00 PM	**ARTIFICIAL INTELLIGENCE LIVE! PANEL DISCUSSION: AI: For Data Scientists or Developers?** *Andrew Brust (moderator), Kevin Feasel, Raj Krishnan, Heather Downing*	
12:00 PM	1:00 PM	Lunch - *Oceana Ballroom*	
1:00 PM	2:15 PM	**AIH06** Google vs. Alexa: Battle of the Bots - *Heather Downing*	**AIH06** Data Orchestration and Data Flow with Azure Data Factory - *Raj Krishnan*
2:30 PM	3:45 PM	**AIH07** To Be Announced	**AIH08** The 8 "C's" of Artificial Intelligence – Being Successful with AI - *Jen Stirrup*
4:00 PM	5:00 PM	Next? Live! 360 Networking Event *Brian Randell, Raj Krishnan, Heather Downing*	
START TIME	**END TIME**	**Artificial Intelligence Live! Post-Conference Workshop: Friday, November 22, 2019**	
7:30 AM	8:00 AM	Registration • Coffee and Morning Pastries	
8:00 AM	5:00 PM	**AIF01** Workshop: From Business Intelligence to Business Analytics with the Microsoft Data Platform - *Jen Stirrup*	

Speakers and sessions subject to change

JAMES MCCAFFREY

Neural Binary Classification Using PyTorch

The goal of a binary classification problem is to make a prediction where the result can be one of just two possible categorical values. For example, you might want to predict the sex (male or female) of a person based on their age, annual income and so on. Somewhat surprisingly, binary classification problems require a slightly different set of techniques than classification problems where the value to predict can be one of three or more possible values.

There are many different binary classification algorithms. In this article I'll demonstrate how to perform binary classification using a deep neural network with the PyTorch code library. The best way to understand where this article is headed is to take a look at the demo program in **Figure 1**.

The demo program creates a prediction model on the Banknote Authentication dataset. The problem is to predict whether a banknote (think dollar bill or euro) is authentic or a forgery, based on four predictor variables. The demo loads a training subset into memory, then creates a 4-(8-8)-1 deep neural network.

After training for 100 iterations, the resulting model scores 98.18 percent accuracy on a held-out test dataset. The demo concludes by making a prediction for a hypothetical, previously unseen banknote. The probability that the unknown item is a forgery is only 0.0215, so the conclusion is that the banknote is authentic.

This article assumes you have intermediate or better programming skills with a C-family language and a basic familiarity with machine learning, but doesn't assume you know anything about binary classification using PyTorch. All of the demo code is presented in this article. The code and the two data files used by the demo are available in the accompanying download. All normal error checking has been removed to keep the main ideas as clear as possible.

Installing PyTorch

PyTorch is a relatively low-level code library for creating neural networks. It's roughly similar in terms of functionality to TensorFlow and CNTK. PyTorch is written in C++, but has a Python language API for easier programming.

Installing PyTorch involves two main steps. First, you install Python and several required auxiliary packages, such as NumPy and SciPy. Second, you install PyTorch as a Python add-on package. Although it's possible to install Python and the packages required to run PyTorch separately, in most cases it's much better to install a

Code download available at msdn.com/magazine/1019magcode.

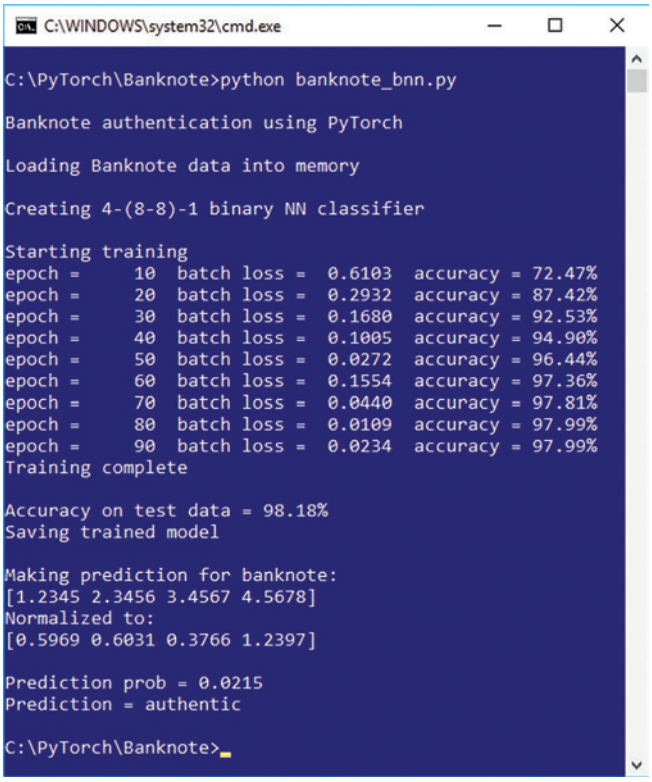


Figure 1 **Binary Classification Using PyTorch**

Python distribution. A distribution is a collection of code libraries containing the base Python interpreter and additional packages that are compatible with each other. For my demo, I installed the Anaconda3 5.2.0 distribution, which contains Python 3.6.5.

> PyTorch is a relatively low-level code library for creating neural networks.

After installing Anaconda, I went to the pytorch.org Web site and selected the options for the Windows OS, pip installer, Python 3.6 and no-GPU version. This gave me a URL that pointed to the corresponding .whl (pronounced "wheel") file, which I downloaded to my local machine. I downloaded PyTorch version 1.0.0. (If you're

new to the Python ecosystem, you can think of a Python .whl file as somewhat similar to a Windows .msi file.) I opened a command shell, navigated to the directory holding the .whl file and entered the command:

```
pip install torch-1.0.0-cp36-cp36m-win_amd64.whl
```

Understanding the Data

The Banknote Authentication dataset has 1,372 items. The raw data looks like:

```
3.6216, 8.6661, -2.8073, -0.44699, 0
4.5459, 8.1674, -2.4586, -1.4621, 0
...
-2.5419, -0.65804, 2.6842, 1.1952, 1
```

The first four values on each line are the predictor values. The last value on each line is either 0 (authentic) or 1 (forgery). The predictor values are from a digitized image of each banknote and include variance, skewness, kurtosis and entropy. All the predictors are numeric. If the data had a categorical predictor such as color, those values could've been converted to numeric values using either 1-of-(N-1) or one-hot encoding.

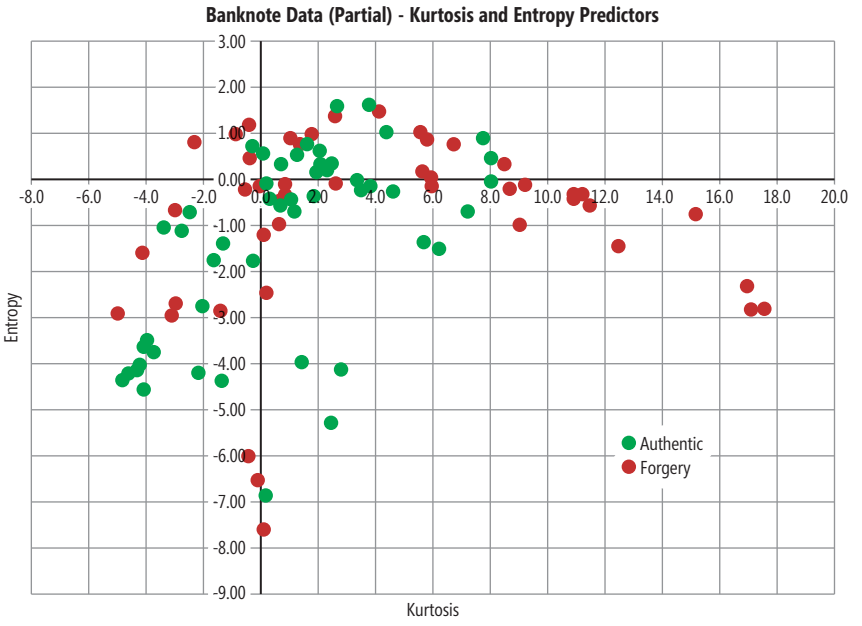


Figure 2 **Partial Banknote Authentication Data**

Because there are four predictor variables, it isn't possible to easily visualize the dataset, but you can get a rough idea of the data from the graph in **Figure 2**. The graph shows the kurtosis and entropy values for the first 100 of the 1,372 data items. Notice that simple linear prediction algorithms would likely perform poorly on this data because it isn't linearly separable.

The first step to prepare the raw data is to randomly split the dataset into a training set and a test set. I split as 80 percent (1097 items) for training and the remaining 20 percent (275 items) for testing. Next, when using a neural network, it's advisable to normalize numeric predictors so that values with large magnitudes don't overwhelm small values. I used min-max normalization on the four predictor variables in the training set.

For each predictor column, I computed the min value and the max value, and then for every value x, normalized as (x - min) / (max - min). After min-max normalization, all values will be between 0.0 and 1.0, where 0.0 maps to the smallest value, and 1.0 maps to the largest value. I saved the min-max values for each column and then normalized the test data using those values. Note that you should normalize test data using the training set min-max values rather than normalize each dataset independently.

During normalization I replaced the comma separators used in the raw data by tab characters. I saved the training and test data in a subdirectory named Data. The demo program code that loads the two datasets into memory is:

```
train_file = ".\\Data\\banknote_norm_train.txt"
test_file = ".\\Data\\banknote_norm_test.txt"

train_x = np.loadtxt(train_file, delimiter='\t',
  usecols=[0,1,2,3], dtype=np.float32)
train_y = np.loadtxt(train_file, delimiter='\t',
  usecols=[4], dtype=np.float32, ndmin=2)
test_x = np.loadtxt(test_file, delimiter='\t',
  usecols=[0,1,2,3], dtype=np.float32)
test_y =np.loadtxt(test_file, delimiter='\t',
  usecols=[4], dtype=np.float32, ndmin=2)
```

Notice that PyTorch wants the Y data (authentic or forgery) in a two-dimensional array, even when the data is one-dimensional (conceptually a vector of 0 and 1 values). The default data type for PyTorch neural networks is 32 bits because the precision gained by using 64 bits usually isn't worth the memory and performance penalty incurred.

The Demo Program

The complete demo program, with a few minor edits to save space, is presented in **Figure 3**. I indent with two spaces rather than the usual four spaces to save space. Note that Python uses the "\" character for line continuation. I used Notepad to edit my program. Most of my colleagues prefer a more sophisticated editor, but I like the raw simplicity of Notepad.

The demo program starts by importing the NumPy and PyTorch packages and assigning shortcut aliases. An alternative to importing the entire PyTorch package is to import just the necessary modules, for example, import torch.optim as opt.

Defining the Neural Network Architecture

The demo defines a 4-(8-8)-1 neural network model with these statements:

```
class Net(T.nn.Module):
  def __init__(self):
    super(Net, self).__init__()
    self.hid1 = T.nn.Linear(4, 8)  # 4-(8-8)-1
    self.hid2 = T.nn.Linear(8, 8)
    self.oupt = T.nn.Linear(8, 1)
...
```

The number of input nodes, four in this case, is determined by the data. For binary classification, by far the most common approach is to use a single output node where a value less than 0.5 maps to class zero (authentic) and a value greater than 0.5 maps to class one (forgery). The number of hidden layers (two in the demo) and the number of nodes in each hidden layer (eight in the demo) are hyperparameters that must be determined by trial and error.

Figure 3 **The Binary Classification Demo Program**

```
# banknote_bnn.py
# Anaconda3 5.2.0 (Python 3.6.5), PyTorch 1.0.0
# raw data looks like:
#  4.5459, 8.1674, -2.4586, -1.4621, 0
#  0 = authentic, 1 = fake

import numpy as np
import torch as T
# -----------------------------------------------------------
class Batcher:
  def __init__(self, num_items, batch_size, seed=0):
    self.indices = np.arange(num_items)
    self.num_items = num_items
    self.batch_size = batch_size
    self.rnd = np.random.RandomState(seed)
    self.rnd.shuffle(self.indices)
    self.ptr = 0

  def __iter__(self):
    return self

  def __next__(self):
    if self.ptr + self.batch_size > self.num_items:
      self.rnd.shuffle(self.indices)
      self.ptr = 0
      raise StopIteration  # exit calling for-loop
    else:
      result = self.indices[self.ptr:self.ptr+self.batch_size]
      self.ptr += self.batch_size
      return result
# -----------------------------------------------------------
def akkuracy(model, data_x, data_y):
  # data_x and data_y are numpy array-of-arrays matrices
  X = T.Tensor(data_x)
  Y = T.ByteTensor(data_y)   # a Tensor of 0s and 1s
  oupt = model(X)            # a Tensor of floats
  pred_y = oupt >= 0.5       # a Tensor of 0s and 1s
  num_correct = T.sum(Y==pred_y)  # a Tensor
  acc = (num_correct.item() * 100.0 / len(data_y))  # scalar
  return acc
# -----------------------------------------------------------
class Net(T.nn.Module):
  def __init__(self):
    super(Net, self).__init__()
    self.hid1 = T.nn.Linear(4, 8)  # 4-(8-8)-1
    self.hid2 = T.nn.Linear(8, 8)
    self.oupt = T.nn.Linear(8, 1)

    T.nn.init.xavier_uniform_(self.hid1.weight)
    T.nn.init.zeros_(self.hid1.bias)
    T.nn.init.xavier_uniform_(self.hid2.weight)
    T.nn.init.zeros_(self.hid2.bias)
    T.nn.init.xavier_uniform_(self.oupt.weight)
    T.nn.init.zeros_(self.oupt.bias)

  def forward(self, x):
    z = T.tanh(self.hid1(x))
    z = T.tanh(self.hid2(z))
    z = T.sigmoid(self.oupt(z))  # necessary
    return z
# -----------------------------------------------------------
def main():
  # 0. get started
  print("\nBanknote authentication using PyTorch \n")
  T.manual_seed(1)
  np.random.seed(1)

  # 1. load data
  print("Loading Banknote data into memory \n")
  train_file = ".\\Data\\banknote_norm_train.txt"
  test_file = ".\\Data\\banknote_norm_test.txt"

  train_x = np.loadtxt(train_file, delimiter='\t',
    usecols=[0,1,2,3], dtype=np.float32)
  train_y = np.loadtxt(train_file, delimiter='\t',
    usecols=[4], dtype=np.float32, ndmin=2)
  test_x = np.loadtxt(test_file, delimiter='\t',
    usecols=[0,1,2,3], dtype=np.float32)
  test_y =np.loadtxt(test_file, delimiter='\t',
    usecols=[4], dtype=np.float32, ndmin=2)

  # 2. define model
  print("Creating 4-(8-8)-1 binary NN classifier \n")
  net = Net()

  # 3. train model
  net = net.train()  # set training mode
  lrn_rate = 0.01
  bat_size = 16
  loss_func = T.nn.BCELoss()  # softmax() + binary CE
  optimizer = T.optim.SGD(net.parameters(), lr=lrn_rate)

  max_epochs = 100
  n_items = len(train_x)
  batcher = Batcher(n_items, bat_size)
# -----------------------------------------------------------
  print("Starting training")
  for epoch in range(0, max_epochs):
    if epoch > 0 and epoch % (max_epochs/10) == 0:
      print("epoch = %6d" % epoch, end="")
      print("  batch loss = %7.4f" % loss_obj.item(), end="")
      acc = akkuracy(net, train_x, train_y)
      print("  accuracy = %0.2f%%" % acc)

    for curr_bat in batcher:
      X = T.Tensor(train_x[curr_bat])
      Y = T.Tensor(train_y[curr_bat])
      optimizer.zero_grad()
      oupt = net(X)
      loss_obj = loss_func(oupt, Y)
      loss_obj.backward()
      optimizer.step()
  print("Training complete \n")

  # 4. evaluate model
  net = net.eval()  # set eval mode
  acc = akkuracy(net, test_x, test_y)
  print("Accuracy on test data = %0.2f%%" % acc)

  # 5. save model
  print("Saving trained model \n")
  path = ".\\Models\\banknote_model.pth"
  T.save(net.state_dict(), path)
# -----------------------------------------------------------
  # 6. make a prediction
  train_min_max = np.array([
    [-7.0421, 6.8248],
    [-13.7731, 12.9516],
    [-5.2861, 17.9274],
    [-7.8719, 2.1625]], dtype=np.float32)

  unknown_raw = np.array([[1.2345, 2.3456, 3.4567, 4.5678]],
    dtype=np.float32)
  unknown_norm = np.zeros(shape=(1,4), dtype=np.float32)
  for i in range(4):
    x = unknown_raw[0][i]
    mn = train_min_max[i][0]  # min
    mx = train_min_max[i][1]  # max
    unknown_norm[0][i] = (x - mn) / (mx - mn)

  np.set_printoptions(precision=4)
  print("Making prediction for banknote: ")
  print(unknown_raw)
  print("Normalized to:")
  print(unknown_norm)

  unknown = T.Tensor(unknown_norm)  # to Tensor
  raw_out = net(unknown)       # a Tensor
  pred_prob = raw_out.item()   # scalar, [0.0, 1.0]

  print("\nPrediction prob = %0.4f " % pred_prob)
  if pred_prob < 0.5:
    print("Prediction = authentic")
  else:
    print("Prediction = forgery")

if __name__=="__main__":
  main()
```

The demo code explicitly initializes the hidden node and output node weights using the Xavier Uniform (also known as Glorot Uniform) algorithm, and initializes the biases to zero. This is the default mechanism so explicit initialization could've been omitted. But in my opinion, it's good practice to explicitly initialize because the default initialization scheme could change in the future.

The demo code specifies the hidden layer and output layer activation functions in the forward function:

```
def forward(self, x):
  z = T.tanh(self.hid1(x))
  z = T.tanh(self.hid2(z))
  z = T.sigmoid(self.oupt(z))
  return z
```

For relatively shallow neural networks, the tanh activation function often works well for hidden layer nodes, but for deep neural networks, ReLU (rectified linear units) activation is generally preferred. The output node has logistic sigmoid activation, which forces the output value to be between 0.0 and 1.0.

The demo program uses a program-defined class, Net, to define the layer architecture and the input-output mechanism. An alternative is to create the network by using the Sequential function, for example:

```
net = T.nn.Sequential(
  T.nn.Linear(4,8), T.nn.Tanh(),
  T.nn.Linear(8,8), T.nn.Tanh(),
  T.nn.Linear(8,1), T.nn.Sigmoid())
```

> Because PyTorch works at a relatively low level of abstraction, there are several different ways to implement each part of a prediction system.

Because PyTorch works at a relatively low level of abstraction, there are several different ways to implement each part of a prediction system. This gives you a lot of flexibility, but increases the difficulty of trying to understand code examples.

Training the Model

Training the model/network is prepared with these eight statements:

```
net = net.train()  # set training mode
lrn_rate = 0.01
bat_size = 16
loss_func = T.nn.BCELoss()
optimizer = T.optim.SGD(net.parameters(), lr=lrn_rate)
max_epochs = 100
n_items = len(train_x)
batcher = Batcher(n_items, bat_size)
```

The learning rate (0.01), batch size (16), and max epochs (100) must be determined by trial and error. For binary classification with a single logistic sigmoid output node, you can use either binary cross entropy or mean squared error loss, but not cross entropy (which is used for multiclass classification). The demo uses a program-defined class Batcher to serve up the indices of 16 training items at a time. An alternative approach is to use the built-in Dataset and DataLoader objects in the torch.utils.data module.

An epoch is one complete pass through all training items. Because there are 1,097 training items and each batch is 16 items, there are 1097 / 16 = 68 weight and bias update operations per epoch. During training, the prediction accuracy of the model is computed and displayed every 10 epochs using a program-defined function named akkuracy. The akkuracy function operates at the Tensor level using efficient aggregate operations. During the development of the demo, I used a function named accuracy that uses a less efficient approach.

Making a Prediction

After the model was trained, the demo used the model to make a prediction for a new, previously unseen banknote. First, the four pairs of min-max values for each predictor variable in the training data are placed into a matrix:

```
train_min_max = np.array([
  [-7.0421, 6.8248],
  [-13.7731, 12.9516],
  [-5.2861, 17.9274],
  [-7.8719, 2.1625]], dtype=np.float32)
```

Recall that the first predictor variable is image variance. So, in the 1,097 training items, the smallest variance is -7.0421 and the largest variance is 6.8248.

The unknown banknote is set to arbitrary values (1.2345, 2.3456, 3.4567, 4.5678) and then min-max normalized, like so:

```
unknown_raw = np.array([[1.2345, 2.3456, 3.4567, 4.5678]],
  dtype=np.float32)
unknown_norm = np.zeros(shape=(1,4), dtype=np.float32)
for i in range(4):
  x = unknown_raw[0][i]
  mn = train_min_max[i][0]  # min
  mx = train_min_max[i][1]  # max
  unknown_norm[i] = (x - mn) / (mx - mn)
```

A PyTorch network expects two-dimensional input (though there are some exceptions), so the demo sets up input with one row and four columns. The prediction is made with these statements:

```
unknown = T.Tensor(unknown_norm)  # to Tensor
raw_out = net(unknown)       # a Tensor
pred_prob = raw_out.item()   # scalar, [0.0, 1.0]
```

The network requires a Tensor object so the NumPy matrix is converted to a Tensor. A quirk of PyTorch is that if a Tensor has a single value, the value can be extracted using the Tensor.item method.

Wrapping Up

The field of neural machine learning is advancing with tremendous speed. Significant new algorithms and neural architectures are appearing every few months. At the time this article was written, three neural network code libraries appear to be distancing themselves from the dozens of those available. PyTorch and TensorFlow are starting to be the most commonly used libraries where some customization or flexibility is needed. The Keras library is becoming the library of choice for situations where a relatively straightforward neural network can be used. But it's too early to predict which of these libraries (if any) will become de facto standards. ■

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 for reviewing this article: *Chris Lee, Ricky Loynd*

LIVE! 360
TECH EVENTS WITH PERSPECTIVE

Cloud & Containers LIVE!

CLOUD-NATIVE, PAAS & SERVERLESS COMPUTING

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

New In 2019!

All Things Cloud-Native

Cloud & Containers Live! has the content you need to understand and leverage Kubernetes, Docker, Azure, AWS, and other cloud-native technologies. Whether you are looking for IT infrastructure, software development, or DevOps information, this conference is for you. Our expert speakers cover topics including cloud-native software design, development, release, deployment, operations, instrumentation/monitoring, and maintenance. We believe the future of most computing is the cloud: containers, PaaS, and serverless. This conference provides you with the knowledge you need to succeed with cloud-based infrastructure and software development.

TRACK TOPICS INCLUDE:

- **Cloud-Native Development**
- **Cloud-Native DevOps**
- **Cloud-Native Infrastructure**

SAVE $300 With Early Bird Savings When You Register by October 25!
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

CCLive360.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

Cloud-Native Development	Infrastructure	Cloud DevOps

START TIME	END TIME		
Cloud & Containers 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	**CCS01** Hands-On Lab: Hands-On with Cloud-Native .NET Development - *Rockford Lhotka*	
2:00 PM	7:00 PM	Pre-Conference Registration - Royal Pacific Resort Conference Center	
START TIME	**END TIME**	**Cloud & Containers 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	**CCM01** Workshop: Building, Running & Continuously Deploying Microservices with Docker Containers on Azure - *Marcel de Vries & René van Osnabrugge*	**CCM02** Workshop: Kubernetes Zero to Hero - Installation, Configuration, and Application Deployment - *Anthony Nocentino*
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**	**Cloud & Containers Live! Day 1: Tuesday, November 19, 2019**	
7:00 AM	8:00 AM	Registration • Coffee and Morning Pastries	
8:00 AM	9:00 AM	**CLOUD & CONTAINERS LIVE! KEYNOTE: To Be Announced**	
9:15 AM	10:30 AM	**CCT01** Serverless with Knative - *Mete Atame*	**CCT02** Intro to Kubernetes - *Chris Kinsman*
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	**CCT03** Fast Focus: Serverless in Azure 101 *- Eric D. Boyd*	**CCT04** Fast Focus: Containers on AWS *- Chris Kinsman*
2:00 PM	2:20 PM	**CCT05** Fast Focus: Deploying Apps to Kubernetes Using Helm *- Rockford Lhotka*	**CCT06** Fast Focus: Building & Debugging Container-based Apps with VS2019 *- Benjamin Day*
2:20 PM	2:45 PM	Networking Break • Visit the EXPO - *Pacifica 7*	
2:45 PM	4:00 PM	**CCT07** Building a Better Distributed Job Scheduler on Kubernetes *- Chris Kinsman*	**CCT08** Lock the Doors, Secure the Valuables, and Set the Alarm *- Eric D. Boyd*
4:15 PM	5:30 PM	**CCT09** Microservices and Containers with Service Fabric and Azure Service Fabric Mesh - *Eric D. Boyd*	**CCT10** Practical Container Scenarios in Azure *- Anthony Nocentino*
5:30 PM	7:30 PM	Exhibitor Reception - *Pacifica 7*	
START TIME	**END TIME**	**Cloud & Containers Live! Day 2: Wednesday, November 20, 2019**	
7:30 AM	8:00 AM	Registration • Coffee and Morning Pastries	
8:00 AM	9:15 AM	**CCW01** Azure App Service Security for ASP.NET Core MVC and WebAPI *- Benjamin Day*	**CCW02** To Be Announced
9:30 AM	10:45 AM	**CCW03** Microservice Architecture *- Rockford Lhotka*	**CCW04** Automate Your Life with PowerShell on Lambda *- AM Grobelny & Steve Roberts*
10:45 AM	11:30 AM	Networking Break • Visit the EXPO - *Pacifica 7*	
11:30 AM	12:30 PM	**LIVE! 360 KEYNOTE: To Be Announced**	
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	**CCW05** Architecting .NET Solutions in a Docker Ecosystem *- Alex Thissen*	**CCW06** To Be Announced
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	**CCW07** Serverless .NET on AWS *- AM Grobelny*	**CCW08** To Be Announced
7:30 PM	9:00 PM	Live! 360 Dessert Luau - *Wantilan Pavilion*	
START TIME	**END TIME**	**Cloud & Containers Live! Day 3: Thursday, November 21, 2019**	
7:30 AM	8:00 AM	Registration • Coffee and Morning Pastries	
8:00 AM	9:15 AM	**CCH01** Busy Developer's Guide to the Clouds *- Ted Neward*	**CCH02** Kubernetes Runtime Security *- Jen Tong*
9:30 AM	10:45 AM	**CCH03** Standup Comedy Hour – Why App Folks Love Infrastructure Folks? *- Vishwas Lele & Jack O'Connell*	**CCH04** Implementing Zero Downtime Application Deployments on Azure PaaS - Marcel de Vries
11:00 AM	12:00 PM	**CLOUD & CONTAINERS LIVE! PANEL DISCUSSION: WHAT THE CLOUD-NATIVE EVOLUTION MEANS TO YOU** *Rockford Lhotka (moderator), Vishwas Lele, Marcel de Vries, Nick Pinheiro, & Jen Tong*	
12:00 PM	1:00 PM	Lunch - *Oceana Ballroom*	
1:00 PM	2:15 PM	**CCH05** Modernize Your App to be Delivered as a SaaS Service *- Nick Pinheiro*	**CCH06** Azure Cloud Transformation Done Right *- Marcel de Vries*
2:30 PM	3:45 PM	**CCH07** Enable External Access to Your Custom Apps with Azure AD B2B *- Nick Pinheiro*	**CCH08** Running 30-Year-Old Software as a Cloud Native SaaS Solution with Docker and Kubernetes on Azure - Roy Cornelissen
4:00 PM	5:00 PM	Next? Live! 360 Networking Event *Rockford Lhotka, Vishwas Lele, & Marcel de Vries*	
START TIME	**END TIME**	**Cloud & Containers Live! Post-Conference Workshop: Friday, November 22, 2019**	
7:30 AM	8:00 AM	Registration • Coffee and Morning Pastries	
8:00 AM	5:00 PM	**CCF01** Workshop: Kubernetes on Azure *- Vishwas Lele*	

Speakers and sessions subject to change

DAVID PLATT

Who're You Looking At?

We tackle many computing problems thinking only of the upside—"Wouldn't it be cool if we could [whatever]?" However, once we've solved the interesting computing problem, we find that its application to society is always double-edged. There's no such thing as a single-edged technology. I think that's the Second Law of Thermodynamics.

The example I've been thinking most about lately is facial recognition. It's a fabulous technical challenge, harnessing all kinds of computing to do what a newborn baby can do within a week of birth. It's tricky, but we've now crossed the watershed to where, while not perfect, it's commercially viable.

The upside of this accomplishment is obvious. Programs and devices can now recognize us when we want them to. Microsoft Hello now automatically recognizes its user, without the need for passwords (see msdn.com/magazine/mt833498). Think of the seconds saved over many interactions with many users, and we've recovered a whole lot of time for humanity.

The downside is also obvious. Programs and devices can now recognize us when we don't want them to. The security systems at the local casino could tell my boss I was there playing blackjack when I claimed to be home sick. Governments and security forces could track and record exactly who attended a rally for which cause. Is this the type of information we want companies and governments stockpiling?

We have to make these decisions soon. The refusal to make them will itself be a definite decision, and probably not the one we want. Microsoft President Brad Smith calls for immediate government action: "We believe it's important for governments in 2019 to start adopting laws to regulate this technology. The facial recognition genie … is just emerging from the bottle," Smith says. "We risk waking up five years from now to find that facial recognition services have spread in ways that exacerbate societal issues. By that time, these challenges will be much more difficult to bottle back up." (See bit.ly/2NwRF7q.)

San Francisco has banned the use of facial recognition by its police force or any other municipal agency (see bit.ly/2ZuEtCc). I understand the concern that leads to this choice. However, the city is deliberately forgoing the benefits it could reap. For example, New Delhi identified 3,000 missing children in its first four-day trial of facial recognition (see bit.ly/2NyMVOL). Is San Francisco doing its best for its citizens if this is now possible, and the city refuses to use it?

The European Union's GDPR governs facial recognition, and just levied its first violation fine. A Swedish high school was using facial recognition to automatically take attendance, with consent from students and their parents (see bit.ly/2Zs9Z84). The commission ruled that true consent was impossible in this situation, given the power imbalance between the parties. Good call? Bad call?

San Francisco has banned the use of facial recognition by its police force or any other municipal agency.

For better or worse, government is how our society makes these kinds of decisions. But I cringe at the thought of technologically illiterate legislators regulating technology. Remember Mark Zuckerberg's testimony in front of the U.S. Senate in April 2018? (You can view the highlights at youtu.be/Egl_KAkSyCw, or enjoy some of the great memes that came out of it at bit.ly/2KVuEcG.)

We, as an industry, are going to have to educate our legislators about what they're voting on. Explaining tech to civilians is what I do—see my book, "Why Software Sucks," and many of these columns. Accordingly, I've offered my services pro bono to my local U.S. Representative, Seth Moulton (D-MA 6th), to act as "tech explainer in chief." I haven't heard back yet. But we're going to need a lot more of this.

I recall Winston Churchill's classic statement that "democracy is the worst form of government except for all those other forms that have been tried from time to time." I think we're about to find out what he meant. ■

Dear readers, you've all heard by now of the impending shutdown of *MSDN Magazine*. I'm thinking of continuing this column on my own after the final issue next month. I've set up a new blog at **davidsplatt.com**, with a column ready for you right now. Please have a look, and sign up for notifications if you would be so kind. If I get 500 subscribers, I'll keep it going. Tell your friends. And your enemies, too. Thanks.

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!

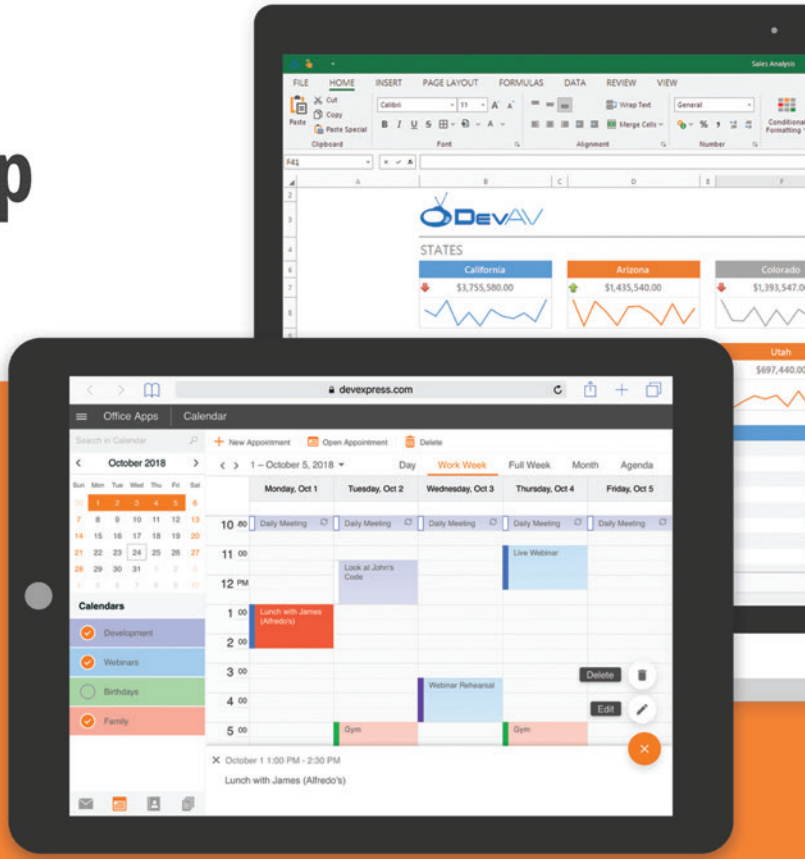
About Esri

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**

DevExpress®

Your Next Great App Starts Here



UI Components
Reporting & Analytics

WIN ASP MVC WPF UWP JS

What's in Issue #105?

Official Supplier

As official and authorized distributors, we supply you with legitimate licenses directly from 200+ software publishers.

24/5 Customer Service

Need help to find the right software license, upgrade or renewal? Call, Email or Live Chat with our experts.

Trusted for Over 20 Years

Over 950,000 licenses delivered to Developers, SysAdmins, Corporations, Governments & Resellers, worldwide.

Open 24 hours a day, Monday to Friday

International Customer Service Centers in the US, UK, Ireland and Japan.

United States	888 850 9911
US Gov't Sales	888 850 9966
United Kingdom	0800 581111
Japan	0120 343 550
Argentina	0800 666 0185
Australia	1 800 0 15292
Austria	0800 281 750
Belgium	0800 7 5603
Brazil	0800 891 6478
Bulgaria	00800 115 4445
Canada	888 850 9911
Chile	1230 020 6857
China (North)	10800 481 1661
China (South)	10800 813 1594
Colombia	01800 710 2043
Croatia	0800 222550
Czech Republic	800 143 313
Denmark	80 88 17 20
Ecuador	1 800 010 562

Finland	0800 1 18002
France	0800 90 92 62
Germany	0800 186 07 06
Greece	00800 44121891
Hong Kong	800 908 581
Hungary	06800 16674
India	000 800 44 01 455
Indonesia	001 803 00811 351
Ireland	1 800 535 661
Israel	180 924 2003
Italy	800 790046
Japan	0120 343 550
Korea, Rep. of	00798 14 800 6332
Malaysia	1 800 81 7261
Mexico	800 681 1559
Netherlands	0800 022 8832
New Zealand	800 449181
Norway	800 1 3685
Panama	00 1 800 203 1587

Peru	0800 53288
Philippines	1800 1816 0315
Poland	00800 442 1092
Portugal	800 844 125
Russia	810 800 2251 1044
Singapore	800 810 2136
Slovenia	0800 80297
South Africa	0800 99 1088
Spain	900 93 8926
Sweden	020 794 989
Switzerland	0800 83 5305
Taiwan ROC	00 801 814313
Thailand	001 800 81 4 5257
Turkey	00 800 4491 3617
United Kingdom	0800 581111
United States	888 850 9911
Uruguay	000 401 902 38
Venezuela	0 800 100 9103
Viet Nam	120 81 863

USA

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

Tel: +1 770 250 6100
Fax: +1 770 250 6199

Europe (UK)

ComponentSource
The White Building
33 King's Road
Reading
Berkshire
RG1 3AR
United Kingdom

Tel: +44 118 958 1111
Fax: +44 118 958 9999

Europe (Ireland)

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

Tel: +353 1 685 6704
Fax: +353 1 685 6725

Asia-Pacific

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

Tel: +81 3 3237 0281
Fax: +81 3 3237 0282

/n software Red Carpet Subscription

Top 50 Product | Top 25 Publisher | Enterprise Partner

A comprehensive suite of Internet components. Any protocol, any platform, any IDE.

Publisher: /n software | Category: Communication & Messaging | ★★★★★

Components for every major protocol from FTP to IMAP to SNMP, SSL and SSH security, S/MIME encryption, Digital Certificates, Credit Card Processing, Bluetooth Low Energy, Internet of Things (IoT), and e-business (EDI) transactions.

Internet Development

IP*Works! provides programmable, SSL-enabled components that facilitate tasks such as ensuring security, sending email, transferring files, managing networks, browsing the Web and consuming Web Services.

Mule Connectors

The /n software Connectors for MuleSoft are pure Java connectors and modules that seamlessly extend MuleSoft's functionality - complete with built-in, Enterprise-ready Web connectivity, secure messaging, and file transfer.

IP*Works! Encrypt

Encrypt and decrypt files, emails, documents, and messages through major cryptographic standards, including S/MIME, OpenPGP, TripleDES, TwoFish, RSA, AES, etc. Complete with X.509 and OpenPGP certificate management.

/n software

Prices from $ 1,535.04

www.componentsource.com/nsoftware-red-carpet-subscription

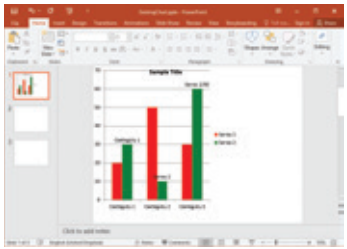
Aspose.Total for .NET

#5 Product | #3 Publisher | Enterprise Partner

Open, create, convert, print and save files from within your own applications.

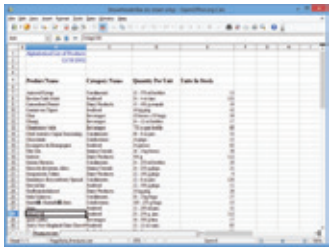
Publisher: Aspose | Category: Document & Text Processing | ★★★★★

Aspose.Total for .NET is a compilation of every .NET component offered by Aspose. It allows you to create a wide range of applications that work with file formats from Excel, Word, PowerPoint, Project, OneNote, Outlook, Visio and PDF.



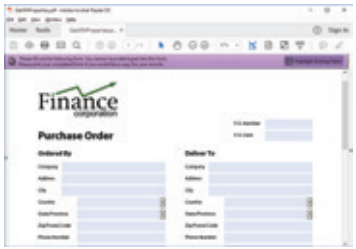
Aspose.Slides for .NET

Enables any .NET application to read, write, modify and convert PowerPoint documents, without the need for Microsoft PowerPoint. It can render presentation slides to PDF, XPS, HTML, images and PDF Notes.



Aspose.Cells for .NET

Enable .NET applications to create, modify, convert, render and print Excel spreadsheets without Microsoft Excel installed on the server. Supports XLS, XLSX, XLSM, XLTX/XLTM, HTML, CSV, ODS, PDF and more.



Aspose.PDF for .NET

Includes a set of APIs for PDF document creation and manipulation, without using Adobe Acrobat. You can also transform each page of a PDF into conventional image formats such as BMP, JPEG, PNG and GIF.

ASPOSE File Format APIs

Prices from $ 2,939.02

www.componentsource.com/aspose-total-net

ASPOSE

File Format APIs

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

US:.....(888) 850 9911
UK:.....0800 581111
France:.....0800 90 92 62
Germany:.....0800 186 07 06
Italy:.....800 790046
Spain:.....900 93 8926

Aspose Product Families

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

- Aspose. Words
- Aspose. PDF
- Aspose. Cells
- Aspose. Email
- Aspose. Slides
- Aspose. Imaging
- Aspose. BarCode
- Aspose. Diagram
- Aspose. Tasks
- Aspose. OCR
- Aspose. Note
- Aspose. CAD
- Aspose. 3D
- Aspose. HTML
- Aspose. GIS
- Aspose. ZIP
- Aspose. Page
- Aspose. PSD
- Aspose. OMR

sales@componentsource.com
www.componentsource.com/aspose

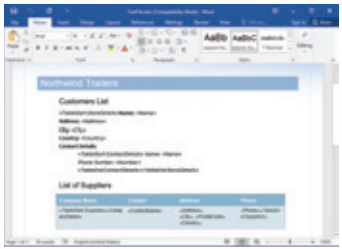
Aspose.Words for .NET

Top 25 Product | #3 Publisher | Enterprise Partner

Advanced .NET APIs to manipulate word processing documents.

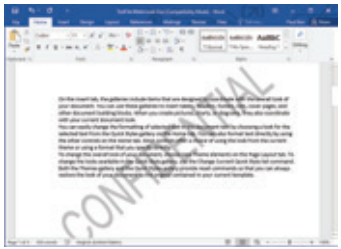
Publisher: Aspose | Category: Word Processing | ★★★★★

Perform a wide range of document processing tasks directly within your .NET applications, without using Microsoft Word. Supports DOC, DOCX, OOXML, RTF, HTML, OpenDocument, PDF, XPS, EPUB and many other formats.



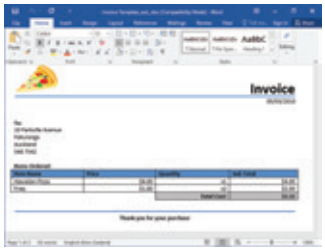
Advanced Document Processing

You can insert formatted text, images, tables and other content into a document. You can also combine documents, move sections between documents, and add images stored outside of the database.



Render, Print & View Word Docs

Print or convert from Word documents to TIFF, PNG, BMP, JPEG, SVG or EMF images. You can also render any document page onto a .NET Graphic object and you can set its size and zoom level to create thumbnails.



Reporting and Mail Merge

Design reports in Microsoft Word and then use Aspose.Words to populate documents with data from a number of sources. You can also perform complex operations including inserting images and repeatable regions.

ASPOSE File Format APIs

Prices from $ 979.02

www.componentsource.com/aspose-words-net

GROUPDOCS

Document Manipulation APIs

APIs to view, convert, edit, annotate, compare, sign, assemble and search documents in your applications.

US:.....(888) 850 9911
UK:.....0800 581111
France:.....0800 90 92 62
Germany:.....0800 186 07 06
Italy:.....800 790046
Spain:.....900 93 8926

GroupDocs Product Families

Manipulate documents from within your own desktop solutions and web apps without requiring any other commercial products.

- GroupDocs. Viewer
- GroupDocs. Annotation
- GroupDocs. Conversion
- GroupDocs. Comparison
- GroupDocs. Signature
- GroupDocs. Assembly
- GroupDocs. Metadata
- GroupDocs. Search
- GroupDocs. Parser
- GroupDocs. Watermark
- GroupDocs. Editor
- GroupDocs. Merger
- GroupDocs. Redaction

sales@componentsource.com

www.componentsource.com/groupdocs

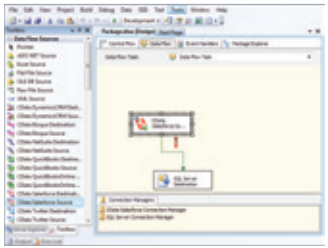
CData SSIS Component Subscription

Top 25 Publisher | Enterprise Partner

Easily connect SSIS Workflows to applications, databases and Web APIs.

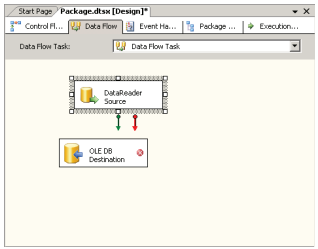
Publisher: CData Software | Category: Data Access Infrastructure

CData SSIS Component Subscription provides an easier and faster way to connect SQL Server with data. It offers straightforward access to NoSQL, Big Data, and SaaS Service data through standard SSIS Workflows.



SQL Server Integration

Use CData Data Flow components and tasks to connect SQL Server with data without expensive custom integration or application development. Includes full Create, Read, Update, and Delete (CRUD) support.



Connect SQL Server with Data

The SSIS Subscription offers access to all CData SSIS components, providing SSIS Workflows with access to applications, databases and Web APIs, in minutes, through SQL Server Integration Services.

Included Data Sources

Integration with 125+ SaaS, NoSQL, & Big Data sources including Amazon SimpleDB, FreshBooks Accounting, Google Apps, Microsoft Dynamics, Office 365, QuickBooks, Salesforce & Force.com and Sugar CRM.

cdata

Prices from $ 979.02

www.componentsource.com/cdata-ssis-component-subscription

Know SQL?

Connect to 125+ data sources (NoSQL, Big Data, SaaS, etc.) through standard SQL queries.

ODBC JDBC ADO.NET SSIS CLOUD BIZTALK EXCEL MOBILE

SQL is the language of data, and our state-of-the-art Drivers let you leverage the full power of SQL to connect with hundreds of applications, databases, and APIs. Through standards-compliant Driver technologies like ODBC, JDBC, ADO.NET, & OData, you can read, write, and update live data, from any data source, using standard SQL queries.

Learn more: www.componentsource.com/cdata

cdata

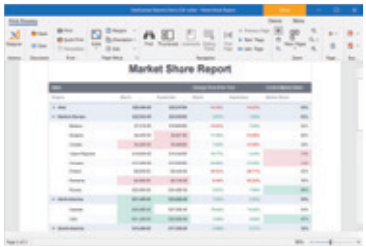
DevExpress Universal

#2 Product | #1 Publisher | Enterprise Partner

Includes over 600 UI Controls, DevExpress Dashboard, eXpressApp Framework and more.

Publisher: DevExpress | Category: Presentation Layer | ★★★★½

Includes charts, data grids, spreadsheets, calendars, schedulers, diagrams, navigation, text editors, PDF, maps and gauges. Supports ASP.NET Web Forms, MVC & Core, WinForms, WPF, UWP, JavaScript, jQuery, Angular and React.



Hierarchical Reports

Hierarchical reports provide built-in Drill-Down support using the DetailBand.DrillDownControl property, clicking a control assigned to this property in Print Preview will expand or collapse the corresponding data level.

ASP.NET Core Rich Text Editor

The Rich Edit control allows you to incorporate advanced text editing functionality, such as formatting and editing, lists, tables, floating objects, document import/export (docx, rtf, txt), mail merge and field support, auto-correct and native client-side printing.

JavaScript Charting

A collection of 30+ responsive charts that enable you to transform data into an eye-catching and elegant visual representation. Includes line, area, bar, pie, funnel, pyramid, financial, range, and polar charts.

DevExpress™

Prices from $ 2,111.99

www.componentsource.com/devexpress-universal

DevExpress

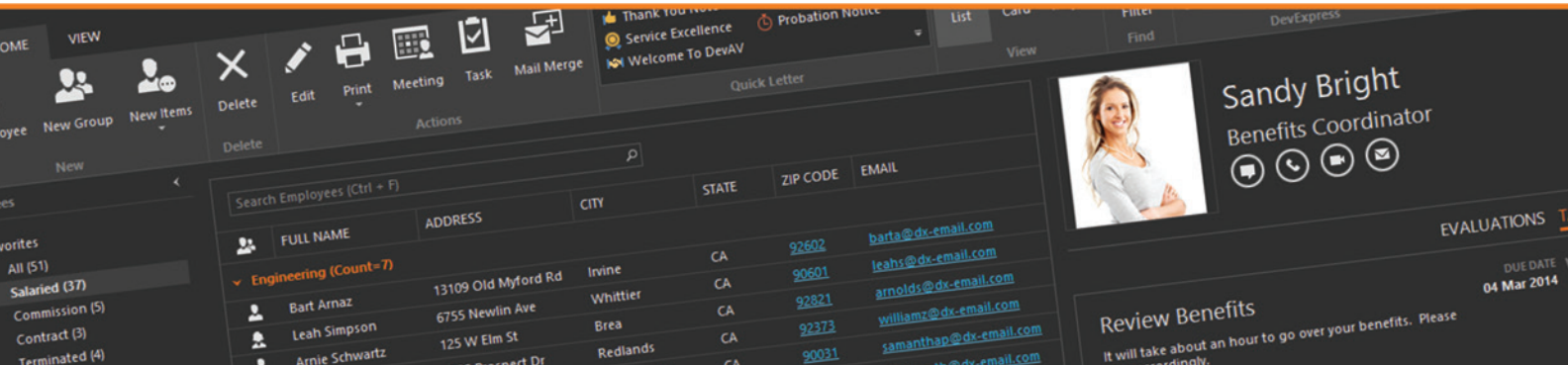
High-Performance Office-Inspired User Interface Controls

DATA GRID | SPREADSHEET | WORD PROCESSING | RIBBON | SCHEDULER | DIAGRAM | AND MORE

Your Next Great Office-Inspired App Starts Here

Learn how you can create high-impact and fully customizable user experiences with our fully integrated suite of UI components and data visualization libraries for Visual Studio.

componentsource.com/devexpress-universal



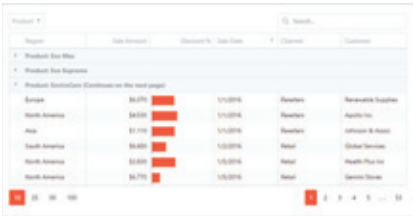
DevExtreme

Top 50 Product | #1 Publisher | Enterprise Partner

HTML5 JavaScript component suite for responsive Web development.

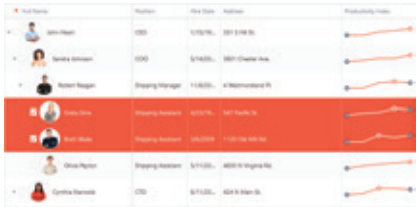
Publisher: DevExpress | Category: Presentation Layer | ★★★★★

Create responsive Web apps for touch devices and traditional desktops. DevExtreme includes data grids, interactive charts, data editors, navigation and multi-purpose widgets that are designed to look great and provide powerful functionality in any browser.



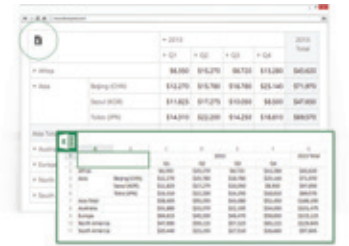
JavaScript Data Grid

With integrated server-side support for data filtering, paging, sorting, record grouping, and summary computations, extremely large datasets are never an issue for the DevExtreme Data Grid.



JavaScript Tree List

An intuitive and easy to use widget that combines the power of a traditional grid and a treeview in a single UI element. From in-memory arrays to JSON, OData or Web API service it can consume data from any information source.



JavaScript Pivot Grid

The DevExtreme Pivot Grid allows you to export its contents to an Excel file with ease. Data types are retained - numbers remain numbers, dates remain dates. This allows users to work with the exported data inside Excel without any data transformations.

DevExpress™

Prices from $ 479.99

www.componentsource.com/devextreme

DevExpress

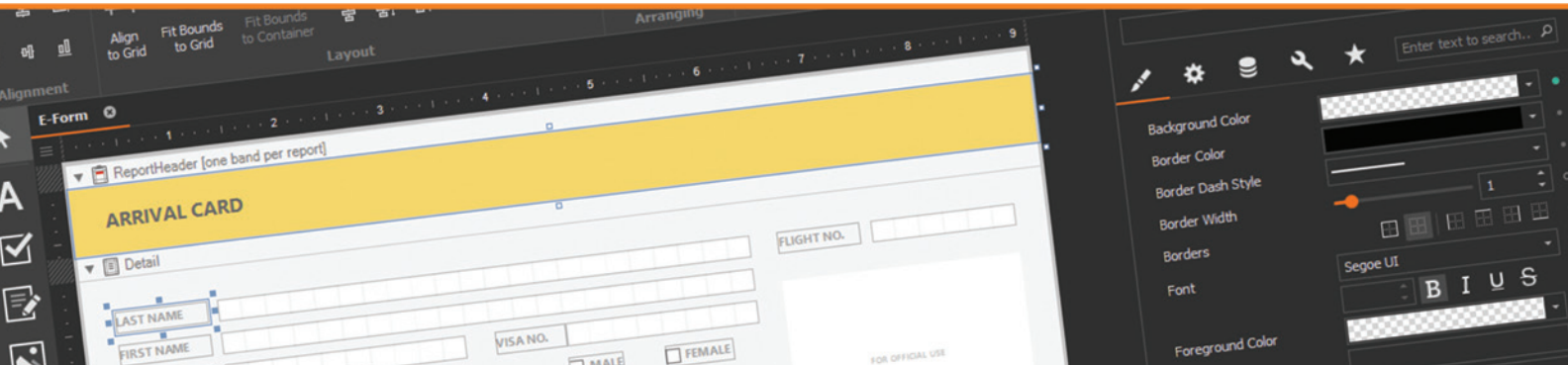
Enterprise-Ready Reporting
for WinForms, WPF, ASP.NET, MVC and .NET Core

WIN WPF ASP MVC .NET CORE

Your Next Great App Starts Here

DevExpress Reporting ships with a fully integrated and easy-to-use
End-User Report Designer for all supported platforms.

componentsource.com/devexpress-universal



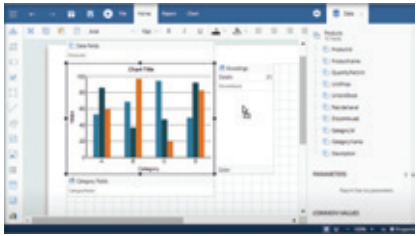
ActiveReports Professional

#3 Product | #2 Publisher | Enterprise Partner

A flexible and extensible platform for every kind of reporting.

Publisher: GrapeCity | Category: Reporting | ★★★★½

Design, customize, publish, and view data in your business applications. Create everything from simple invoices to complex statistical data visualizations. You can even enable your end users to design reports with the ad hoc report designer.

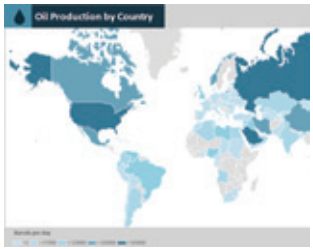


Reporting for End Users

End users always ask for tweaks in standard reports. Give them the power to make the changes themselves with a fully customizable set of designer controls that includes a drag and drop report designer.

Dynamic Reports

You can control report behavior by using events or by designing dynamic reports entirely in code. You can also use property expressions to change the run-time behavior of controls in RDL and Page reports.



.NET Reporting Components

ActiveReports includes a complete set of reporting components and code libraries. Use built-in controls such as barcodes, tablix, charts, data bars, sparklines, and maps to visualize data.

GrapeCity

Prices from $ 1,575.02

www.componentsource.com/activereports-developer

GrapeCity | Developer Solutions

ActiveReports

.NET Reporting Platform for Any Business Need

- Create interactive ad-hoc reports
- Distribute reports to all devices, royalty-free
- Customize with the extensive .NET API
- Integrate charts with data visualization controls
- Schedule and manage enterprise reporting with the reports server
- Design and publish with web based designers and portals

Learn more at **componentsource.com/activereports**

ComponentOne Studio Enterprise

Top 10 Product | #2 Publisher | Enterprise Partner

Award-winning .NET controls for business applications.

Publisher: GrapeCity | Category: Presentation Layer | ★★★★½

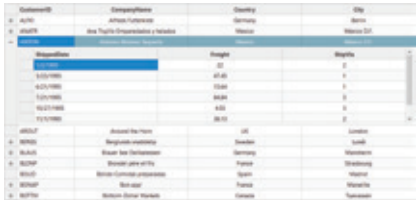
A complete collection of award-winning performant, extensible .NET UI controls for mobile, Web, and desktop. Includes controls for WinForms, WPF, UWP, ASP.NET MVC, Xamarin, JavaScript and server-side APIs for processing images, Excel files and PDFs.

Powerful Data Visualization

Whether you need 60+ charts, Gantt views, pivot tables, gauges, maps, or sparklines, ComponentOne Studio's data visualization controls handle large data sets and impress your users with professional design.

Engaging User Interactions

The UI and navigational control set offers intuitive displays of information with several forms of interaction. Edit, sort, delete, build menus and toolbars, and easily apply themes when crafting tailored UIs.



Industry-Leading FlexGrid

Get power, flexibility, and speed in a single lightweight grid control. Sort, group, filter, merge cells, freeze columns, and import or export to Excel with FlexGrid, an industry-leading flexible data grid.

GrapeCity

Prices from $ 1,472.58

www.componentsource.com/componentone-studio

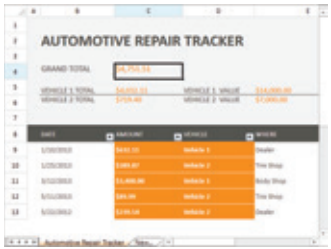
Spread.NET

#4 Product | #2 Publisher | Enterprise Partner

Flexible Excel-like .NET spreadsheet components.

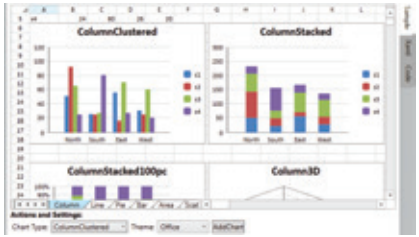
Publisher: GrapeCity | Category: Spreadsheet | ★★★★★

Quickly deliver Microsoft Excel-like experiences with desktop designer apps, create Enterprise spreadsheets, grids, dashboards, and forms with the comprehensive API.



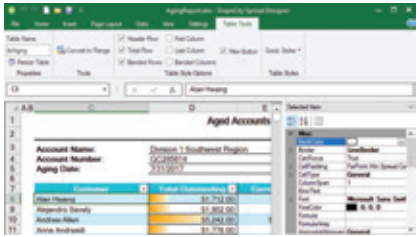
Calculation Engine

Choose from over 450 functions and apply them to individual cells, rows/columns, or entire sheets. Take advantage of cross-sheet referencing, structured formulas, array formulas and custom functions.



Import/Export Excel Files

No spreadsheet is complete without full Microsoft Excel import/export support. Import your most advanced Excel spreadsheets and then export your Spread.NET spreadsheet to XLSX or other file formats, all with zero dependencies on Excel.



No-code design with Spread Designers

Instantly design and apply a powerful Microsoft Excel-like .NET spreadsheet with the desktop designer apps - no learning curve required. Designers are available for WinForms, WPF, and ASP.NET.

GrapeCity

Prices from $ 1,476.52

www.componentsource.com/spread-net

GrapeCity | Developer Solutions

Spread.NET | SpreadJS

.NET and JavaScript Spreadsheet Components

- Deliver true Excel-like spreadsheet experiences, fast
- Create analysis, data collection and management, scientific, and financial applications
- Use for spreadsheets, advanced grids, dashboards, reports, input forms and more
- Powerful calculation engine includes 450+ functions
- Import and export native Excel spreadsheets with no Excel dependencies

Learn more at **componentsource.com/spread**

Infragistics Ultimate

Top 50 Product | Top 10 Publisher | Enterprise Partner

UX design and Enterprise app development for Web, desktop and mobile.

Publisher: Infragistics | Category: Presentation Layer

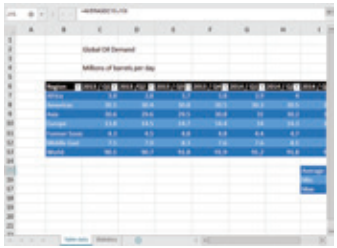
Includes 100+ beautifully styled, high-performance grids, charts, & other UI controls, plus visual configuration tooling, rapid prototyping, and usability testing. Includes components for Angular, JavaScript, ASP.NET, Windows Forms, WPF & Xamarin.

Charts for ASP.NET

Render a broad range of rich chart types from simple bar, line, area, and pie charts to more complex financial, candle, and radar charts. Ensure the best presentation of the chosen chart type with a custom wizard for designing single-layer or multi-layer charts.

React Data Grid

This lightweight Data Grid was built to meet the challenge of displaying large amounts of data while providing versatility and performance on mobile devices. With fast load time and smooth scrolling, you can seamlessly scroll through an unlimited number of rows and columns.



Export & Import from Excel

Enable your users to work with their advanced data. Users can programmatically import data and display it in the Spreadsheet component that supports all Microsoft Excel formulas, structural features, workbooks, and worksheets.

INFRAGISTICS

Prices from $ 1,955.10

www.componentsource.com/infragistics-ultimate

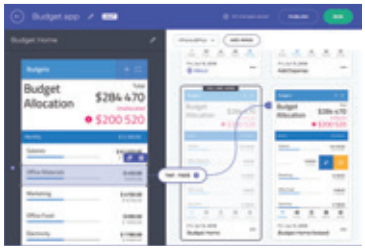
Indigo.Design

Top 10 Publisher | Enterprise Partner

Unparalleled productivity for teams. Pixel-perfect Angular code from Sketch designs.

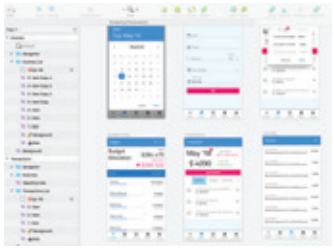
Publisher: Infragistics | Category: Software Architecture & Design

Designer meets developer with Indigo.Design, a unified platform for visual design, UX prototyping, code generation, and app development.



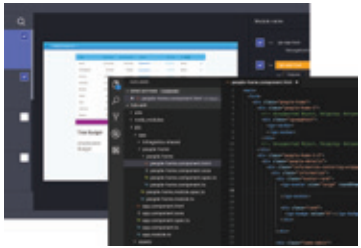
Share, Collaborate & Test

Import Sketch files into the cloud and share user flows and interactions on any device. Record & playback usability studies and get real-time reporting & analytics to see how users interact with your designs.



Design to Match Your Brand

Create best-in-class UI designs using the expressive Indigo Design System with Sketch UI Kits. With 50+ components, 30+ UI patterns and complete app scenarios, getting started is a breeze.



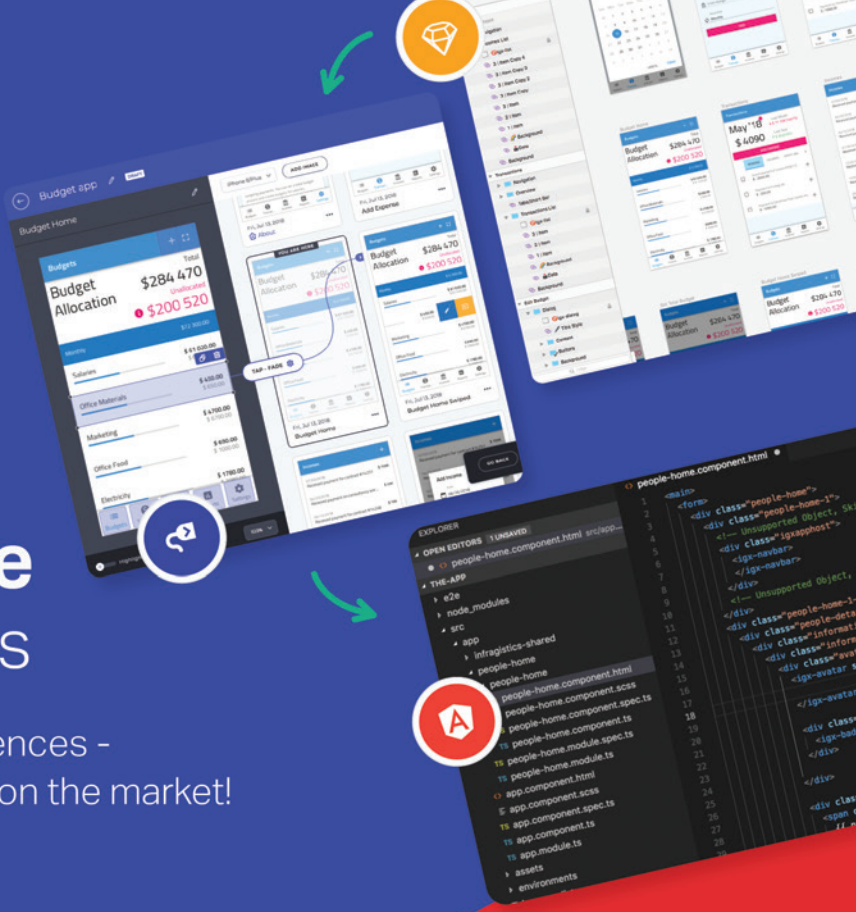
Get Runnable Code with 1-Click

Generate high-quality HTML, CSS, and Angular code from your design with the click of a button. 50+ UI components in Sketch map to the Ignite UI toolset.

INFRAGISTICS

Prices from $ 975.10

www.componentsource.com/indigo-design



Get Angular Code
From Sketch Designs

Design and build modern web experiences -
including the fastest Grids and Chart on the market!

INFRAGISTICS

indigo.design

More information at:
componentsource.com/infragistics

InstallAware Studio

Top 50 Publisher

Enterprise Partner

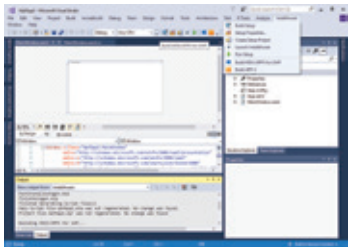
An installer with next generation MSIX Packaging Format support.

Publisher: InstallAware | Category: Development & Deployment

Build Win32, Win64, and .NET apps as MSIX packages, crossing the Microsoft Desktop Bridge to the Microsoft Windows Store.

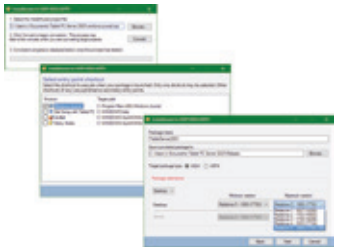
Build Triple-Hybrid AMD64/ ARM64/X86 Installers

InstallAware lets you build a single setup targeting three platforms from one file: AMD64/ EM64T, ARM64, and X86.



Latest Windows 10 Version

Full support for the latest Windows 10 version, including .NET 4.8, C++ 15.9, Visual Studio 2019 and Visual Studio Team Services integration and a MSIX Desktop Bridge for your Desktop apps.



Build MSIX/APPX packages from Visual Studio

A single click on the InstallAware toolbar for Visual Studio builds your active solution/project as an MSIX package.

InstallAware

Prices from $ 1,959.02

www.componentsource.com/installaware-studio

NEW!
InstallAware X9

InstallAware X9 is the first and only installer that you may use to build triple-hybrid installers with X64 (AMD64/EM64T), ARM64, and X86 payloads all delivered from a single setup file!

InstallAware

ARM64 (QUALCOMM SNAPDRAGON 835/50) CELLULAR PC
First and only with full-stack support to build, deploy, and repackage apps.

NEXT GENERATION MSIX PACKAGING FORMAT
Build MSIX packages from the Visual Studio Toolbar, InstallAware IDE, or the command line!

UNIVERSAL CLIENT-FREE DATABASE REFRESH
MS SQL versions through 2017, MySQL versions through 8.x, Oracle versions through 18c.

FULL WINDOWS 10 19H1 / 1903 SUPPORT
Supports .NET 4.8, C++ 15.9, Visual Studio 2019 and the entire eco-system.

ADVANCED 32-BIT COMPATIBLE COMPRESSION
Reduce the already-compressed sizes of .NET 4.52 by 17%, MS SQL Express 2016 with SP1 by 38%!

InstallAware is the most flexible platform for traditional and agile development teams creating Windows and Azure software installers, as well as MSIX Universal Windows Platform, Microsoft App-V Virtualization, and agentless/royalty-free InstallAware Virtualization packages.
Bridge all of your Win32, Win64, and .NET apps through Microsoft's Desktop Bridge to the Microsoft Windows Store using InstallAware's new MSIX Builder!

Visual Studio Partner

www.componentsource.com/installaware

InstallAware MSIX Editor

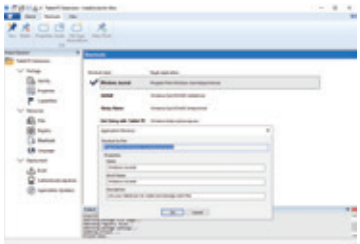
Top 50 Publisher

Enterprise Partner

A full featured IDE for Microsoft's MSIX package format.

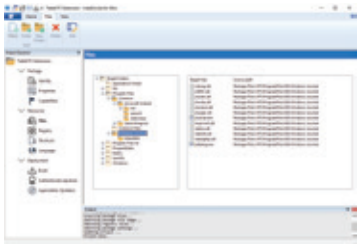
Publisher: InstallAware | Category: Development & Deployment

Instantly load any existing MSIX package, browse its contents and edit it. Then you can save your customized MSIX package directly, or as a MSIX modification package.



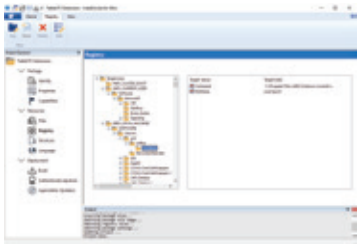
Edit MSIX Packages

You can save your changes as a brand new MSIX package or as a new MSIX modification package. Then you can submit your MSIX packages to the Windows Store.



Load Existing MSIX Packages

Edit MSIX installation packages, even if you don't have their source. View and extract resources from inside of any MSIX package and update the contents and logic.



Deliver Apps via MSIX

Unlike MSI, apps delivered via MSIX run in their own secure sandbox, and still interface with almost as many user endpoints as available in a traditional MSI package.

InstallAware

Prices from $ 489.02

www.componentsource.com/installaware-msix-editor

About InstallAware

InstallAware Software, founded in 2003, is the leading Cloud Infrastructure Company with its laser sharp focus on bullet-proof enterprise software deployment. InstallAware has been recognized by multiple awards from Microsoft, SDTimes "Leader of the Software Development Industry", Visual Studio Magazine Reader's Choice, ComponentSource, WindowsITPro, among others.

InstallAware X9 is available in a complimentary edition for all Visual Studio users and paid editions. For more information, visit https://www.componentsource.com/installaware

- **A SINGLE SETUP FOR X64, ARM64, AND X86 PAYLOADS**
- **MSIX BUILDER FOR NEXT-GENERATION PACKAGING**
- **UNIVERSAL, CLIENT-FREE DATABASE SUPPORT**
- **FULL WINDOWS 10 19H1 / 1903 SUPPORT**
- **ADVANCED 32-BIT COMPATIBLE COMPRESSION**

www.componentsource.com/installaware

Syncfusion Essential Studio Enterprise

Top 100 Product | Top 25 Publisher | Enterprise Partner

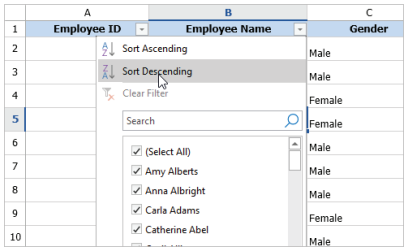
High-performance toolkits for your development projects.

Publisher: Syncfusion | Category: Presentation Layer | ★★★★★

Includes more than 1,000 components and frameworks for WinForms, WPF, ASP.NET (Web Forms, MVC, Core), UWP, Xamarin, JavaScript, Angular, Blazor, Vue and React.

ASP.NET Core UI Controls

Includes over 65 high-performance, lightweight, modular, and responsive UI controls. All the controls are touch friendly and render adaptively based on the device, providing optimal user experience on phones, tablets, and desktops.



WPF Spreadsheet Control

An Excel-inspired control that allows you to create, edit, view, and format Microsoft Excel files without having Excel installed. It includes a built-in calculation engine with more than 400 widely used formulas.

JavaScript DataGrid

A feature-rich control for displaying data in a tabular format. Its wide range of functionalities include data binding, editing, Excel-like filtering, custom sorting, aggregating rows, selection, and support for Excel, CSV, and PDF formats.

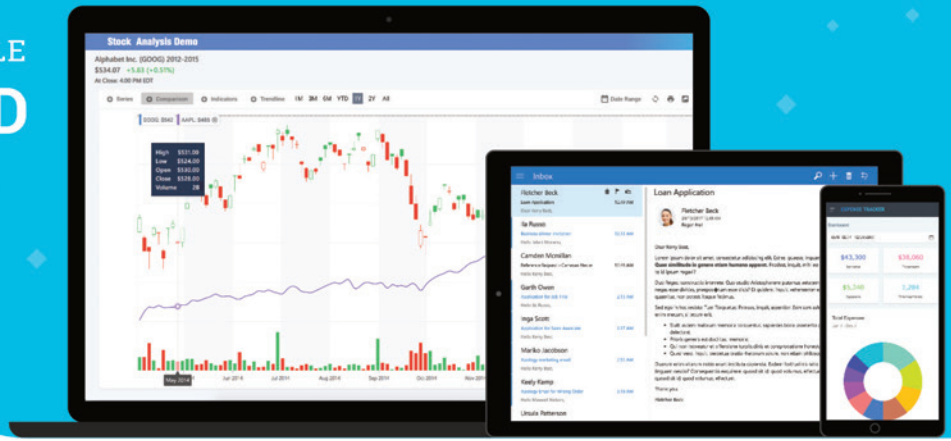
Syncfusion

Prices from $ 2,245.50

www.componentsource.com/syncfusion-essential-studio-enterprise

MORE THAN **1000** CUSTOMIZABLE

CONTROLS AND FRAMEWORKS



ESSENTIAL STUDIO ENTERPRISE EDITION

WEB

- ASP.NET MVC
- ASP.NET Web Forms
- ASP.NET Core
- JavaScript
- Angular
- React
- Vue

MOBILE

- JavaScript
- Xamarin
- Universal Windows Platform

DESKTOP

- Windows Forms
- WPF
- Universal Windows Platform

FILE FORMATS

- Excel
- PDF
- Word
- PowerPoint

DATA SCIENCE

- Predictive Analytics

CHOOSE WITH CONFIDENCE

5

120+
FIVE STAR REVIEWS

Syncfusion

NOW AVAILABLE / Multi-license bundles!

FOR MORE INFORMATION VISIT

componentsource.com/syncfusion

Syncfusion Essential Studio for JavaScript

Top 25 Publisher | Enterprise Partner

A complete JavaScript UI controls library.

Publisher: Syncfusion | Category: Presentation Layer | ★★★★☆

Includes high-performance, lightweight, modular, and responsive UI controls. All the controls are touch friendly and render adaptively based on the device, providing an optimal user experience on phones, tablets, and desktops.

React Pivot Table

Organize and summarize business data and display the result in a cross-table format. Includes data binding, drilling up and down, Excel-like filtering and sorting, editing, Excel and PDF exporting, several built-in aggregations, pivot table field list, and calculated fields.

Angular Word Processor

Provides all common Word processing features including editing text, formatting contents, resizing images and tables, finding and replacing text, bookmarks, tables of contents, printing, and importing and exporting Microsoft Word documents.

Blazor Components Library

65+ UI and data visualization components including DataGrid (editing, grouping and Excel-like filtering), Charts (30+ charts and graphs ranging from line to financial charts) and Scheduler (display multiple calendars in a single layout with finely grouped events).

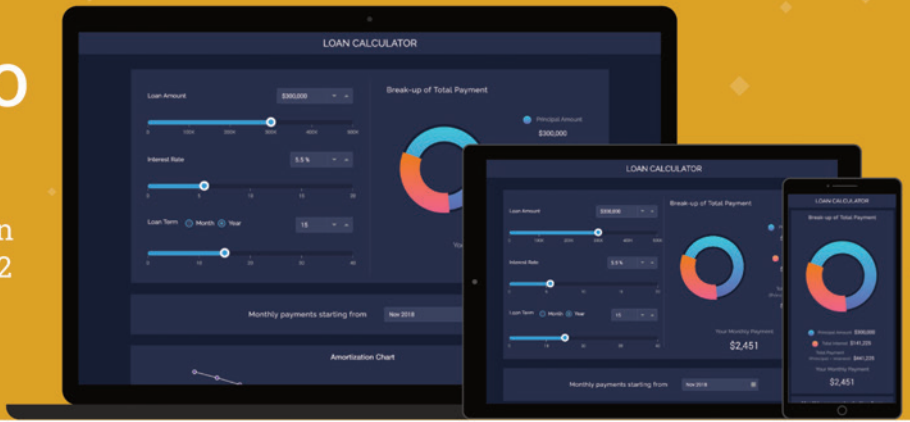
Syncfusion

Prices from $ 975.10

www.componentsource.com/essential-studio-for-javascript

SYNCFUSION ESSENTIAL STUDIO FOR JAVASCRIPT

Featuring Syncfusion's next-generation pure JavaScript package, Essential JS 2



COMPREHENSIVE. MODERN. FREE OF EXTERNAL DEPENDENCIES.

- Lightweight
- Modular
- Responsive
- Customizable with built-in themes
- Built for performance
- Usable in multiple languages

INTEGRATES WITH TOP FRAMEWORKS

- Angular
- React
- Vue

Syncfusion

FOR MORE INFORMATION VISIT

www.componentsource.com/essential-studio-for-javascript

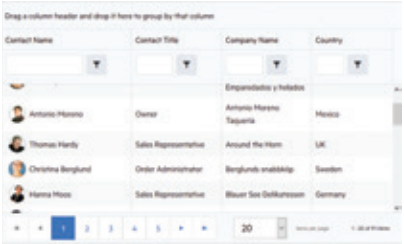
Kendo UI

Top 100 Product | Top 10 Publisher | Enterprise Partner

JavaScript UI components for responsive Web and data visualization.

Publisher: Telerik | Category: Presentation Layer | ★★★★☆

A collection of JavaScript UI components with libraries for jQuery, Angular, React, and Vue. Quickly build eye-catching, high-performance, responsive Web applications - regardless of your JavaScript framework choice.



Grid for Angular

The Kendo UI Grid for Angular displays data in a tabular format and comes with 100+ built-in features, including PDF and Excel exporting, paging, sorting, filtering and data binding. You can also use a context menu in the Grid by integrating ContextMenu for Angular.

Provides Advanced UI Features

Kendo UI includes advanced data grid components, charts, spreadsheets, schedulers, and many more. Customizable themes enable you to effortlessly deploy a consistent look and feel across your apps.

Modern Design & Themes

Kendo UI ensures that any developer can make their applications follow the latest and greatest standards when it comes to design. With implementations of Material Design, Bootstrap, and other modern designs, a stunning UI can be achieved by simply including a CSS file.

Progress Telerik

Prices from $ 881.02

www.componentsource.com/kendo-ui

Telerik DevCraft Complete

Top 25 Product | Top 10 Publisher | Enterprise Partner

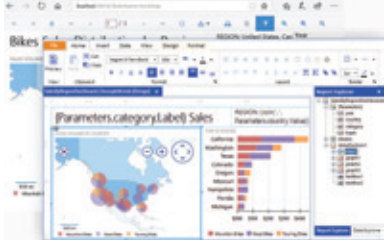
Quickly build modern, high-performance apps for any platform.

Publisher: Telerik | Category: Presentation Layer | ★★★★☆

Build .NET and JavaScript apps with a sleek, fast and consistent UI across all Web, desktop & mobile platforms. DevCraft Complete includes mocking & reporting functionality, and priority support with unlimited support tickets and 24 hour response time.

Document Processing Libraries

Quickly add advanced document processing functionality to your app. In a few clicks your app can create, import, modify, export and print to and from common text, spreadsheet, and PDF file formats.



Reporting

Create and style interactive, reusable, touch-friendly reports in Visual Studio - or in a standalone report designer, deliver them to any mobile, Web or desktop .NET application and print them in more than 20 formats.

UI for Blazor

Blazor gives you the ability to write rich web apps with C# rather than JavaScript. Telerik UI for Blazor components have been built from the ground-up to ensure you experience shorter development cycles, quick iterations and cut time to market.

Progress Telerik

Prices from $ 1,469.02

www.componentsource.com/telerik-devcraft-complete

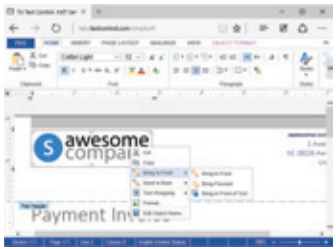
TX Text Control .NET Server for ASP.NET

Top 50 Publisher | Enterprise Partner

Powerful word processing and reporting for your Web Forms & MVC Web applications.

Publisher: Text Control | Category: Word Processing

Specifically designed to run in server side applications, TX Text Control .NET Server for ASP.NET is ideal for batch processing or printing large volumes of documents.



Cross-Browser Document Editing

Includes a true WYSIWYG, HTML5-based Web editor and reporting template designer. Give your users an MS Word compatible editor to create powerful reporting templates anywhere - in any browser on any device.

Automate MS Word Documents

Replacing MS Office Automation in applications is a typical use case for document APIs such as TX Text Control. Automate, edit and create documents using UI and non-UI components.

Document Conversion

Convert and modify different document types. Supports a wide range of word processing formats (RTF, DOC, DOCX, HTML, XML, PDF) and image file formats (GIF, PNG, JPG, BMP, WMF, EMF, TIF).

TEXT CONTROL

Prices from $ 2,938.04

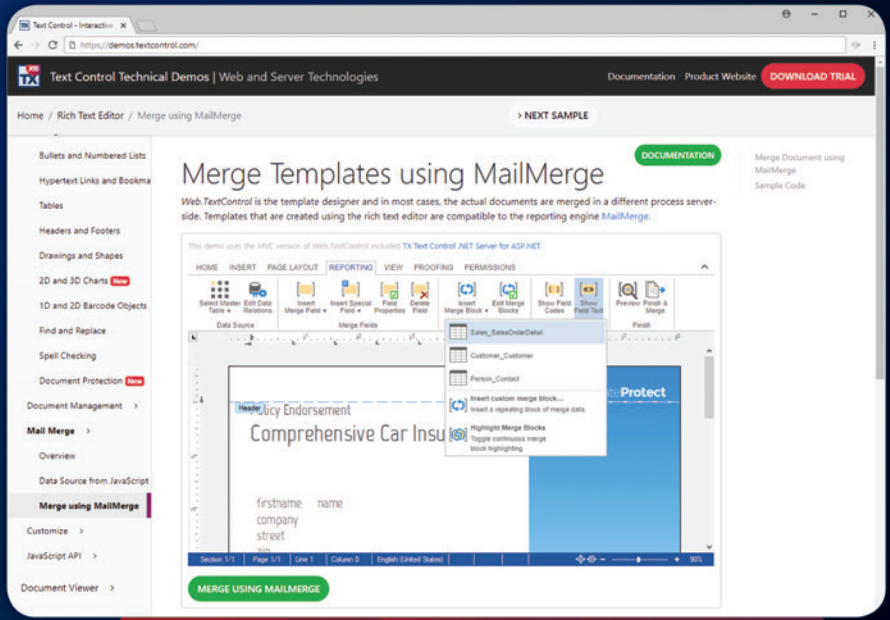
www.componentsource.com/tx-text-control-net-server

TEXT CONTROL

Visual Studio Partner

Integrate Documents and Reporting in Any Platform

The Text Control Reporting Framework combines powerful reporting features with an easy-to-use, MS Word compatible, word processor.



Learn more:

componentsource.com/textcontrol

WE ARE CHANGING THE WAY YOU LOOK AT REPORTING

Text Control ReportingCloud

Top 50 Publisher

Enterprise Partner

Web API reporting platform to create MS Word compatible reports in the Cloud.

Publisher: Text Control | Category: Cloud Services

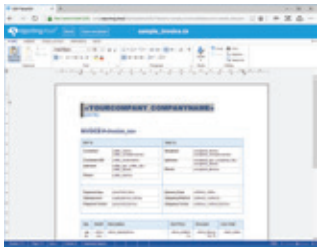
Use the Text Control ReportingCloud Web API to merge MS Word compatible templates with JSON data from all clients such as .NET, JavaScript, PHP, Node.JS, jQuery, Python, Android, Java, iOS and many more.

Merge with Data

Using simple HTTP requests, merge fields and repeating blocks in templates can be merged with hierarchical JSON data from within your application.

Retrieve Created Document

The merged templates are returned as Adobe PDF, MS Word and HTML format and you can use them immediately in your application.



Design your Template

Create customized invoices, contracts and quotes directly using the WYSIWYG online template designer.

TEXT CONTROL

Prices from $ 341.04

www.componentsource.com/text-control-reportingcloud

reportingcloud

TEXT CONTROL

Create Documents in the Cloud from Any Platform

RESTful Web API powered reporting platform to create MS Word compatible reports.

Visual Studio Partner

php

JS

WE ARE CHANGING THE WAY YOU LOOK AT REPORTING

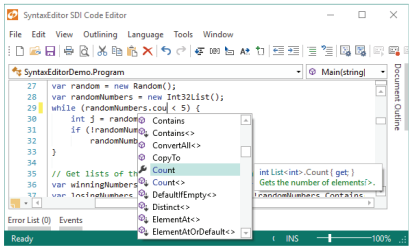
Actipro WPF Studio

Top 100 Product | Top 50 Publisher | Corporate Partner

Everything you need to add rich functionality to your WPF apps.

Publisher: Actipro Software | Category: Presentation Layer | ★★★★★

Actipro WPF Studio includes over 100 WPF controls including barcodes, charts, datagrid, docking & MDI, editors, gauges, micro charts, navigation, property grid, ribbon, syntax editor, themes, tree controls, views, wizards, and a shared library of controls.

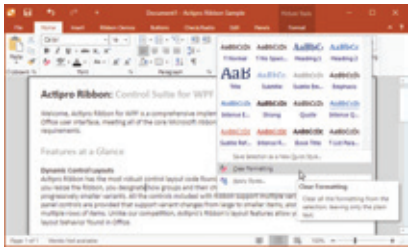


SyntaxEditor for WPF

Add advanced code editing functionality to your WPF apps. Over 20 sample languages are included to get you started (such as C#, HTML, JavaScript, and more), and optional premium add-ons are also available.

Easily Generate Rich Charts

Actipro Charts supports many chart types from basic line and bar charts to complex stacked area charts. It includes useful features such as multiple series, labels, legends, stacking, and customizable palettes.



Ribbon Control

Use the Ribbon control to reproduce an Office-like user interface with split buttons, galleries and mini-toolbars. Various layout controls govern where items are placed within a ribbon as it decreases in width.

actipro

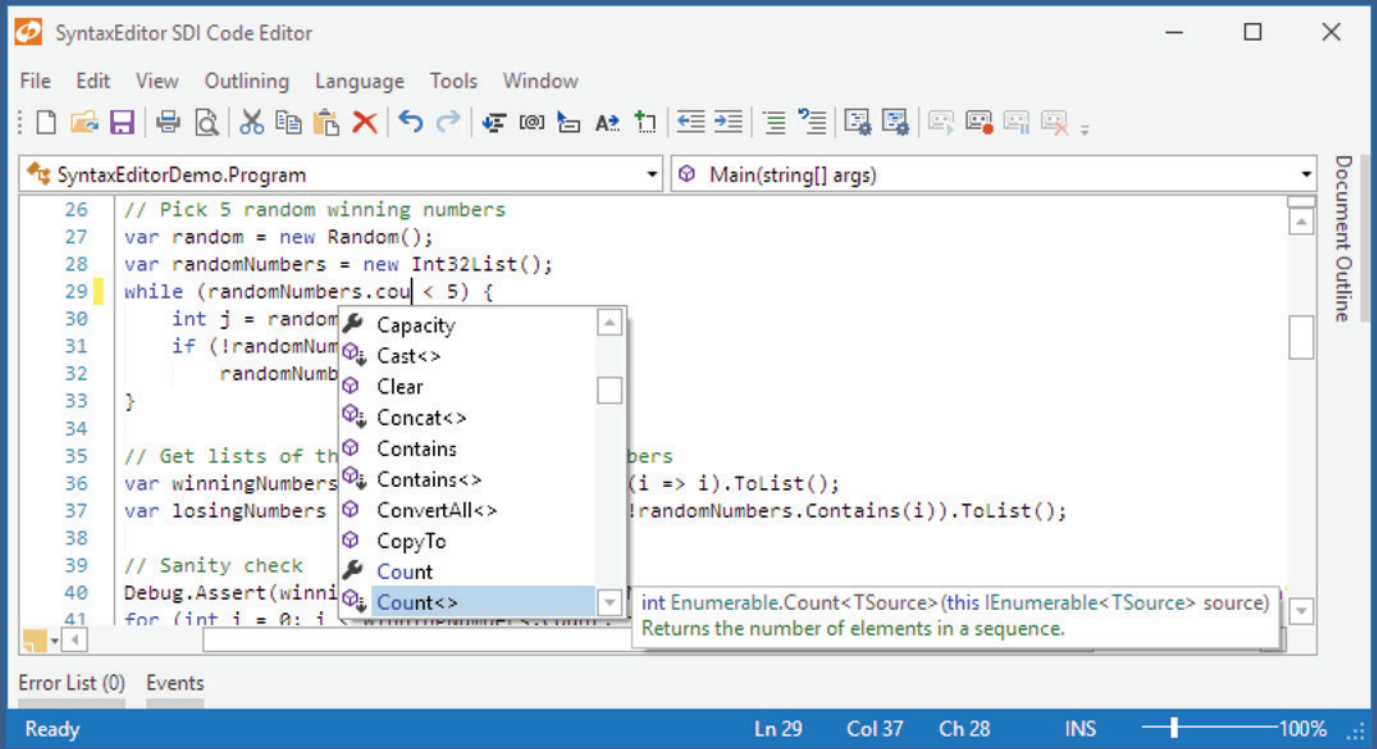
Prices from $ 636.02

www.componentsource.com/actipro-wpf-studio

Actipro SyntaxEditor

for WPF, Universal Windows, Silverlight, WinForms

The ultimate syntax-highlighting code editor control



Discover the possibilities...

componentsource.com/actipro

actipro

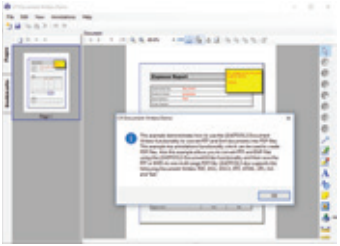
LEADTOOLS Document Imaging Suite SDK

Top 25 Publisher | Corporate Partner

Develop powerful document imaging applications.

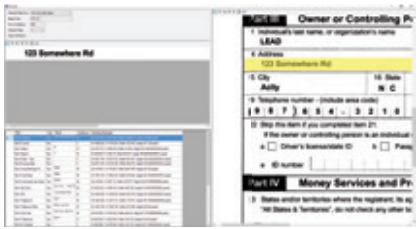
Publisher: LEADTOOLS | Category: Image Editing & Processing | ★★★★★

Build end-to-end document imaging solutions that include OCR, OMR, barcode, forms recognition and processing, PDF, conversion, annotation, HTML5 Zero-footprint viewing, print capture and image viewing functionality.



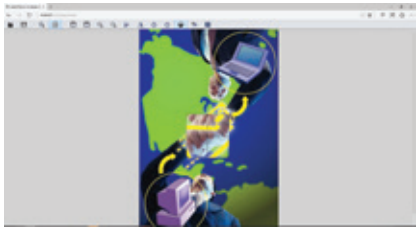
Document Writers SDK

A powerful and flexible development tool that converts a wide variety of document image sources into document formats such as PDF and PDF/A, XPS, ePUB and MOBI, ALTOXML, Microsoft Word (DOC/DOCX), Microsoft Excel (XLS), HTML, RTF, and Text.



Forms Recognition

In addition to basic forms recognition with static field locations, LEADTOOLS is able to detect and process unstructured and semi-structured documents such as invoices, driver's licenses, and passports.



JavaScript Image Viewer

Create zero-footprint, cross-platform applications with document, medical, and raster image viewing functionality. Developers can target desktops, tablets, and mobile devices such as iPad, iPhone, and Android.

LEADTOOLS

Prices from $ 4,895.10

www.componentsource.com/leadtools-document-imaging-suite

LEADTOOLS®

Build Better Apps With LEADTOOLS

LEADTOOLS toolkits will help you create the most powerful desktop, web, server, and mobile applications with the greatest collection of programmer-friendly and cross-platform document, medical, and multimedia technologies.

OCR | BARCODE | PDF | DICOM | OFFICE FORMATS | ANNOTATIONS | VIEWERS | + MANY MORE

Microsoft .NET | macOS | iOS

COMPONENTSOURCE.COM/LEADTOOLS

LEAD TECHNOLOGIES INCORPORATED

Nevron Vision for .NET

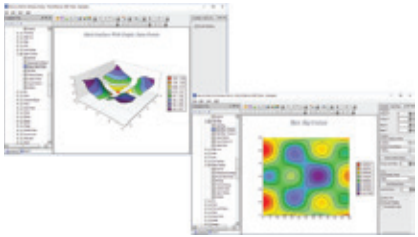
Top 100 Publisher

Corporate Partner

Data visualization components for desktop and Web applications.

Publisher: Nevron | Category: Presentation Layer

A set of charting and diagramming components for applications targeting WinForms, WPF, ASP.NET, and MVC. Features Chart, with 170+ charting types, Diagram, with numerous shapes and automatic layouts, along with many other advanced components.

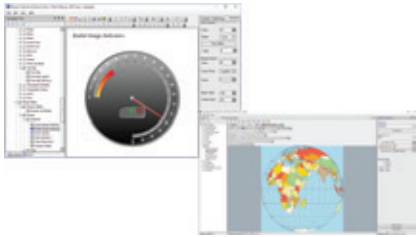


Nevron Chart for .NET

Display virtually any 2D, 3D chart or gauge with superior quality. Packed with hundreds of intuitive examples and many advanced features, Nevron Chart allows you to get started quickly using no code at all.

Nevron Diagram for .NET

A complete diagramming solution packed with interactive features, shapes, automatic layouts, stunning visual effects and comes equipped with ready to use controls to boost your application development.



Nevron Gauge and Map for .NET

Suitable for any application that needs to visualize KPIs, Scorecards or Geographical data. The controls feature a full set of Radial and Linear gauges, LED displays, State Indicators, and map projections.

NEVRON

Prices from $ 827.64

www.componentsource.com/nevron-net-vision

NEVRON

Nevron Data Visualization and UI Controls

The leading data visualization and UI components for desktop and server applications since 1998.

Built-in Map Projections

Text Processing

Presentation Charts

Real Time Charts

2D and 3D Hardware Accelerated Rendering

Organization Diagrams

Maps with ESRI Import

Scientific Charts

Financial Charts

Gauges and KPIs

Automatic Graph Layouts

solutions available for

Microsoft ASP.net | WPF | WinForms | SharePoint | Microsoft SQLServer | Xamarin

Learn more at **componentsource.com/nevron** today

Visual Studio Partner

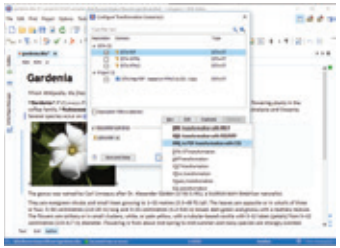
Oxygen XML Editor Enterprise

A complete solution for XML development and authoring.

Top 25 Product | #4 Publisher | Corporate Partner

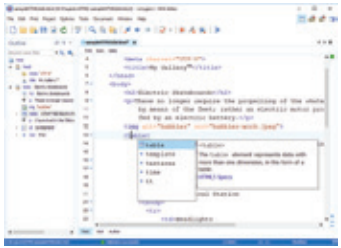
Publisher: Syncro Soft | Category: Structured Document Authoring

A collection of must have tools for XML editing, with support for all types of XML documents and other file types, including XML Schemas, CSS, XSLT, WSDL, RelaxNG, Schematron, Ant, XQuery, and many more.



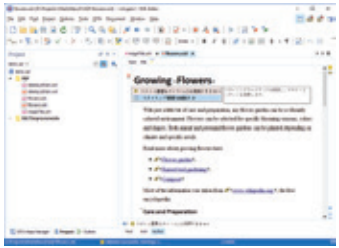
Single-Source Publishing

The XML Editor offers preset and configurable scenarios that are one click away, allowing you to produce outputs in PDF, ePUB, HTML, and many other formats using the same source.



XML Publishing Frameworks

Oxygen offers CSS-based, visual editing support for a number of important XML documentation frameworks (DITA, DocBook, TEI, XHTML). Also, if you are planning to use other types of XML documents, an API is available for customizing Oxygen.



Intelligent XML Editor

XML editing is easier than ever with the help of intelligent actions and easy to use features. Oxygen Quick Fix helps you resolve errors in an XML document by offering quick fixes to problems such as missing required attributes or invalid elements.

syncro soft

Prices from $ 945.45

www.componentsource.com/oxygen-xml-editor-enterprise