ASP.NET Core, Vue.js, Amazon Web Services, Power Query

CODE
JUL
AUG
2020

codemag.com - THE LEADING INDEPENDENT DEVELOPER MAGAZINE - US $ 8.95 Can $ 11.95

CODE

Transform Your ASP.NET Core API into AWS Lambda Functions

Exploring What's New in Vue 3

Building WPF Apps with Custom Scripts

Working with Managed Identity

0 7>
7 25274 56130 8

CODE SECURITY

shutterstock_rangizzz/Africa Studio/onsight

INCREASE YOUR CYBER SECURITY DEFENSES

IDENTIFY POTENTIAL SECURITY PROBLEMS

CODE Security offers application security reviews to help organizations uncover security vulnerabilities. Let us help you identify critical vulnerabilities in complex applications and application architectures and mitigate problems before cybercrime impacts your business.

- **CODE Security** offers three types of security testing for many applications and services, including simulating a real-world attack scenario. We can test web applications, internal applications, mobile applications and do code audits and test hardware. We can even reverse engineer software if you no longer have the source code.
- After our security audit, our **CODE Security** experts provide a technical report including all identified vulnerabilities along with a severity rating according to industry standards.

ADDRESS POTENTIAL SECURITY PROBLEMS

With a list of identified vulnerabilities, what do you do next? We have two suggestions:

- Give your internal developer team the knowledge to address security problems in your applications. **CODE Security** offers two flavors of Secure Coding for Developers courses. Each course is three days but follow the same basic outline. One course is primarily for C# developers and the other is for Java developers. See our training page for a list of upcoming classes.
- If you cannot spare developers to overhaul your software, hire **CODE Consulting** to work with your teams and we'll address the security vulnerabilities together.

Let 2023 be the year you mitigate cyber security risk in your applications.

CODE Security experts can help you identify and correct security vulnerabilities in your products, services and your application's architecture. Additionally, CODE Training's hands-on **Secure Coding for Developers** training courses educates developers on how to write secure and robust applications.

Contact us today for a free consultation and details about our services.

codemag.com/security

832-717-4445 ext. 9 • info@codemag.com

Features

Columns

Departments

US subscriptions are US $29.99 for one year. Subscriptions outside the US pay $49.99 USD. Payments should be made in US dollars drawn on a US bank. American Express, MasterCard, Visa, and Discover credit cards are accepted. Bill Me option is available only for US subscriptions. Back issues are available. For subscription information, send e-mail to subscriptions@codemag.com or contact Customer Service at 832-717-4445 ext. 9.

Subscribe online at www.codemag.com

CODE Component Developer Magazine (ISSN # 1547-5166) is published bimonthly by EPS Software Corporation, 6605 Cypresswood Drive, Suite 425, Spring, TX 77379 U.S.A. POSTMASTER: Send address changes to CODE Component Developer Magazine, 6605 Cypresswood Drive, Suite 425, Spring, TX 77379 U.S.A.

LEADTOOLS® v21

BUILD BETTER APPS WITH LEADTOOLS

Recognition Engine
• OCR, MICR, OMR & ICR • 1D & 2D Barcode • Form, Check, Invoice, Passport, ID & Business Card Recognition

Document Engine
• Viewer • Converter • Document Compare • Document Analyzer • Office Formats • PDF • Document Image Cleanup • Annotations/Markup • Virtual Printer • Scanning • Compression

Medical Engine
• DICOM • PACS • DICOM Web & Desktop Viewer • DICOM Security • HL7 • CCOW • Print to PACS • 3D • Annotations

Imaging Engine
• Load, Save & Convert +150 Formats • Compression • Viewers • Scanning

Multimedia Engine
• Stream • Video Conference • Playback • Conversion • Capture • Audio/Video Codecs • Audio/Video Processing Filters • Multiplexers and Demultiplexers • Directshow • Media Foundation

Microsoft® .NET | Xamarin | Windows | Java | Linux | iOS | macOS | Android | HTML5

Get Started Today

DOWNLOAD OUR FREE EVALUATION

LEADTOOLS.COM

From the Publishers, Editors, and Staff of CODE Magazine

A Moment of Silence

#blacklivesmatter

Nevron Open Vision (NOV)

The future of .NET UI development is here!

NOV lets you build applications
for Windows, Mac and soon Blazor
using a single codebase.

The suite features industry-leading components
including Chart, Diagram, Gauge, Grid,
Scheduler, Rich Text Editor, Ribbon, and others.

Nevron also provides advanced solutions for:

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

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

Managed Identity

Let me introduce you to something that we all use and we all hate: passwords. You'd think this problem is recent, but it isn't. In World War 2, Germans had a tool called the enigma machine, the main purpose of which was encrypted communication. The military version had a switchboard in the front. The two communicating parties had to agree ahead of time on the pattern that they'd use. Pattern, password: same thing, different smell. In reality, and in the daily struggle of war, which, frankly, our jobs feel like sometimes, they got sick of changing passwords, and they just stuck with one easy-to-guess password. Guess how they ended each message. With "Heil Hitler." Those two bits of information made it possible for the allies to decrypt their messages. Imagine how much damage they sustained because they were too lazy to change a password?

Sahil Malik
www.winsmarts.com
@sahilmalik

Sahil Malik is a Microsoft MVP, INETA speaker, a .NET author, consultant, and trainer.

Sahil loves interacting with fellow geeks in real time. His talks and trainings are full of humor and practical nuggets.

His areas of expertise are cross-platform Mobile app development, Microsoft anything, and security and identity.

Fast forward to today. You still have organizations insisting on super complex passwords that must be changed every two months. Some of my utility companies force me to do that. It's mind boggling. My only recourse is to write it down somewhere or I'll simply forget it. So where do we write it down? A password manager! Yes, what a great idea, let's collect all of our valuable passwords, protected by yet another password.

I fully realize that attempts are being made to get around this password drama, but let's be honest, we are far from it. There may be technical solutions available, but have you entered a password anywhere earlier today?

Passwords suck! They are hard to remember, they are hard to secure, they expire, rotating them is a pain, etc., etc.

But then, a big part of our architecture includes headless processes. Or programs that need to access resources securely. For instance, something trying to access a password manager of sorts. Let's say that in Azure there's something called a Key Vault, which lets you manage secrets and certificates. And your code wishes to access Key Vault securely.

You could, of course, access it as an interactive user. But how would that code work with DevOps? You'd have to put the password in some config file, I guess. Or maybe a refresh token, which is, well, almost like a password. And then you'd have to somehow make sure that you don't accidentally check it into source control.

Looks like we are back to square one.

Wouldn't it be nice if you had something that gave all the advantages and possibilities of having an identity for your running code, without the headache of credential management?

That very thing is managed identity.

What Is Managed Identity?

Managed identity is a feature of Azure Active Directory that lets you assign an identity to various Azure resources, without the headache of managing the identity's credential. You can use this identity to authenticate to any service that supports Azure AD authentication, such as Microsoft Graph, Key Vault, custom APIs, etc.

As you may know, when you protect an API using Azure AD, you're doing so by validating an incoming access token. For instance, when you make a call to Microsoft Graph, Microsoft Graph, being yet another API, expects you to include an access token in the authorization header. This token must be for the audience Microsoft Graph. When making such a call using managed identity, such a token must be for an audience of Microsoft Graph (or any other API you wish to target).

There is one other key important point. When making calls to protected APIs, you have the ability to make such a call using delegated permissions or application permissions. Delegated permissions, as you may know, require the user identity to be present. Application permissions, on the other hand, require only the application's identity to be present. Managed identities can only make calls that use application permissions.

There are two kinds of manage identities available in Azure AD. A system-assigned managed identity and a user-assigned managed identity.

A **system-assigned managed identity** is enabled directly on an Azure service instance. When the identity is enabled, Azure creates an identity for the instance in the Azure AD tenant that's trusted by the subscription of the instance. After the identity is created, the credentials are provisioned onto the instance. The lifecycle of a system-assigned identity is directly tied to the Azure service instance that it's enabled on. If the instance is deleted, Azure automatically cleans up the credentials and the identity in Azure AD.

A **user-assigned managed identity** is created as a standalone Azure resource. Through a create process, Azure creates an identity in the Azure AD tenant that's trusted by the subscription in use. After the identity is created, the identity can be assigned to one or more Azure service instances. The lifecycle of a user-assigned identity is managed separately from the lifecycle of the Azure service instances to which it's assigned.

If you read between the lines, there's a key advantage to using a user-assigned managed identity. That advantage is that you can manage the lifecycle of the managed identity separately from the Azure resources that the identity is assigned to. In other words, not only can you share that identity across multiple Azure resources, but perhaps a bigger advantage is that you know the identity involved ahead of time, since it wasn't auto-provisioned for you. This means that you can set up the permissions, RBAC (role-based access control) assignments, etc., ahead of time. Now, this may be anecdotal, or just one person's opinion, but I find system-assigned managed identity is great for demos and user-assigned managed identity is great for real world scenarios.

Another very key player in the managed identity world is the Azure instance metadata service or IMDS for short. IMDS is a REST endpoint accessible to all IaaS VMs created via the Azure Resource Manager. The endpoint is available at a well-known non-routable IP address (169.254.169.254) that can be accessed only from within the VM. Whenever any code running on a virtual machine with a managed identity needs to ask for an access token for the managed identity, instead of asking the AzureAD endpoints, it asks for their token from this IMDS endpoint. Although this reduces the workload on Azure AD endpoints, this itself this has two significant downsides.

- IMDS will give you a cached view of what it thinks Azure AD knows about this identity. This cache view may be a little out of date. This cache, at the time of writing this article, is up to eight hours old, although efforts are underway to reduce this latency.
- Any program running on the virtual machine has equal rights to this IMDS endpoint. In other words, you can't say within one virtual machine that you have two managed identities, one for each running program. If you need two identities, you will need two virtual machines.

Now, I use the term virtual machine very loosely here. In fact, managed identities in Azure apply to way more than just virtual machines. You can assign a managed identity to a docker image, to an Azure function, to app services, or to a virtual machine, etc. Additionally, a number of services in Azure AD understand incoming managed identity. You can find a list of the current services here https://docs.microsoft.com/en-us/azure/active-directory/managed-identities-azure-resources/services-support-managed-identities. Undoubtedly, this list will continue to increase as time moves forward. Managed identity is a very key investment for Azure AD.

How Do Managed Identities Work?

Managed identity is a service principal of a special type reserved for use with Azure resources. The key difference between a managed identity service principal and a normal service principal is that when a managed identity is created, the managed identity resource provider issues a certificate internally to the identity. Now your code can freely use the managed identity to request access tokens for services that support Azure AD authentication. You never have to worry about rolling the credentials, because that's the responsibility of Azure. The default period for credential rotation is 46 days, but the individual resource provider can choose to wait longer if necessary. At this point, when your code wishes to get an access token using the managed identity, it simply requests it from the IMDS endpoint at http://169.254.169.254/metadata/identity/oauth2/token.

Managed Identity Calling Microsoft Graph

Next let's wrap up your understanding with a real-world example. Frequently in our applications, we end up writing services, also known as Cron jobs. These are applications that don't have a user interface. They're like services that run behind the scenes doing important stuff, stuff that cannot be done synchronously and must be handled by a service.

Usually, when you'd like to have a service perform call an API securely for you, you get an access token using something called a client credential flow. Client credential flow, by definition, requires you to present a credential. This credential can either be a password or it can be a certificate. And with that, you're back to the headache of managing that credential.

Managed identities give you the advantages of a real identity without the headache of managing a credential. In this example, I'm going to show you how you can use managed identity, working as a daemon, accessing Microsoft Graph.

Microsoft Graph has a huge API surface. I'm going to show you a very simple example of how to call the endpoint that gives the details of all users in the tenant. In order to do, so I'll have to go through some steps. I'll have to create a managed identity. I'll have to assign that managed identity to something that understands that managed identity; in this case, I'll use a function. I'll have to write some code that's able to get an access token for Microsoft Graph on behalf of this managed identity. And, of course, I'll need to make sure that my managed identity has the permissions to call Microsoft Graph.

As you can see, it doesn't matter what I'm calling in Microsoft Graph, or, for that matter, what I'm calling on any API protected by Azure Active Directory. The steps are identical. From what I'm about to present here in this article, you should be able to extrapolate in calling any Azure Active Directory protected API.

The Main Steps

In order for my example to work, let's outline the main steps I need to accomplish. There are five main steps.

1. Create the function app in Azure.
2. Create a user-assigned managed identity.
3. Assign the managed identity to the function app.
4. Grant permissions to the managed identity to call Microsoft Graph.
5. Author the function app that runs on a timer, gets an access token, and calls MSGraph.

It's worth mentioning that everything I'm about to show here will work with the system-assigned managed identity as well. However, I've chosen to go with a user-assigned managed identity because I feel that it's a little bit more practical, a bit more real world.

Additionally, I choose to show you this code example in NodeJS. Although I must emphasize that any of these concepts are platform-agnostic. Everything I'm showing here will work in Python, .NET Core, or any other language you prefer. And because I'm going to use functions as my managed identity host, you need to make sure that you work with something that functions can support.

Finally, a function app can be triggered by a number of possible triggers. The easiest possible implementation is the HTTP trigger, because all the project templates are designed to give you an HTTP trigger very easily. I'll keep it a little more real-world by using a timer trigger. This is because I intend to use this managed identity as a serverless process, and a timer job makes the most sense here.

Enough background. Let's get our hands dirty with some code.

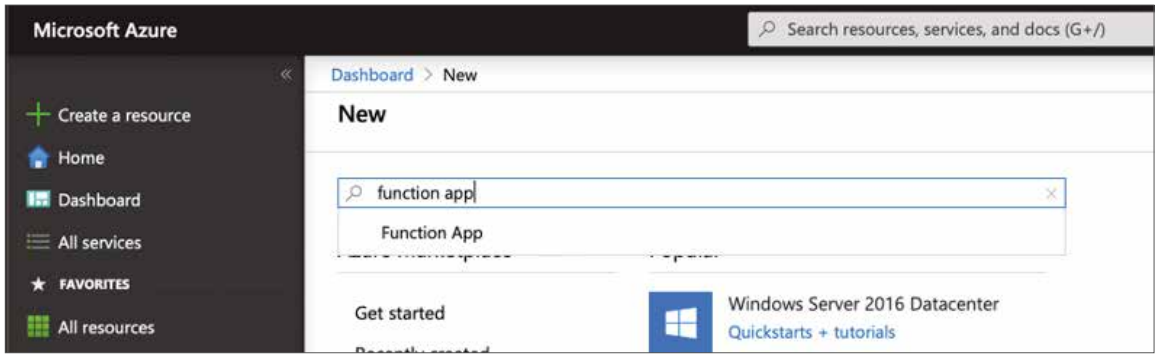


Figure 1: The Function App template

Create the Function App in Azure

This part is easy. Just go to Azure and create a new function app. To do so, simply search for **function app** and click the Create button. This can be seen in **Figure 1**.

The Azure portal with them shows you a user interface, like that shown in **Figure 2**. Go ahead and provide the details. As you can see, my function app is called "sahiltimerapp" and it runs on NodeJS version 12.

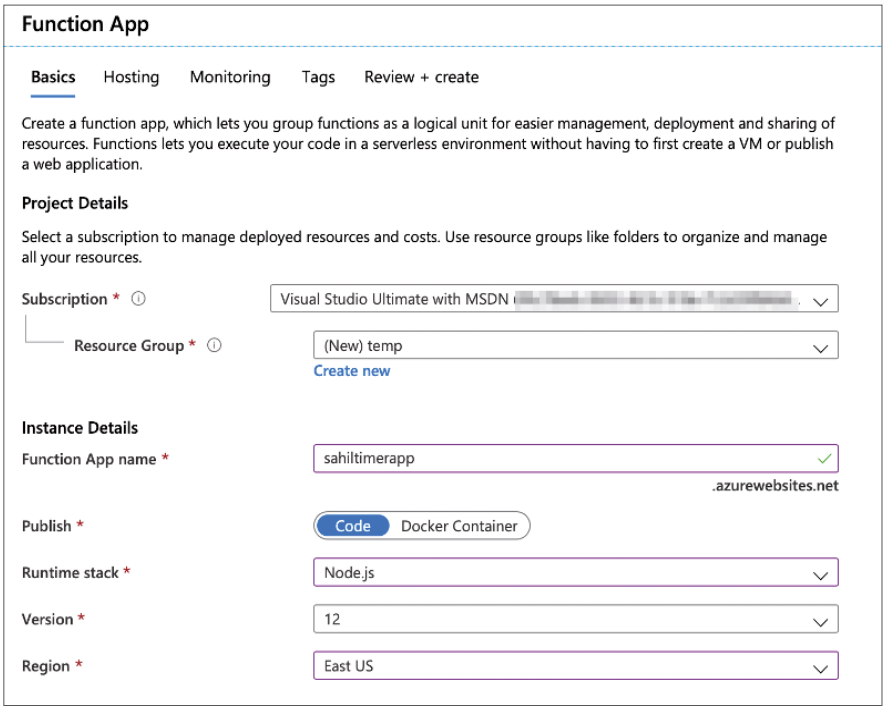


Figure 2: Function App details

Figure 3: Creating a user-assigned managed identity

Create a User-Assigned Managed Identity

This is also pretty simple. Just search for "user-assigned managed identity" in Azure Portal, as can be seen in **Figure 3**. Of course, you may also choose to create this from Azure CLI or PowerShell or Microsoft Graph. There are many ways to achieve this goal.

When creating a user-assigned managed identity, you will be asked to provide a name for it. I called my managed identity sahiltimerfunctionidentity.

Once you provide all the details and create the managed identity, in the Azure Portal, go to its properties, and get its Client ID and Object ID. You'll need both in a moment. Mine looks like **Figure 4**.

Assign the Managed Identity to the Function App

You have a function app and you have a managed identity. It's now time to assign this newly created managed identity to your function app. This is quite simple to do. In the Azure portal, navigate to your function app. And under its properties, go to platform features. Over there, look for identity, as shown in **Figure 5**.

Under the identity section of the function app, choose to assign the user-assigned managed identity to the function app, as shown in **Figure 6**.

Grant Permissions to the Managed Identity to Call Microsoft Graph

Unfortunately, at the time of writing this article, there's no easy user interface built inside of the Azure portal to grant permissions to a managed identity. Note that you can view the permissions, but you can't grant the permissions through the user interface at this time. Additionally, there's no direct commandlet in Azure CLI to achieve this either.

Currently, there are two ways to grant permissions to a managed identity to any arbitrary API. One is using PowerShell, and other is using Microsoft Graph. Because Microsoft Graph is a cross-platform approach and I'm on a Mac, I choose to show you how to do this using Microsoft Graph.

To work with Microsoft Graph, I'm going to need an access token. There are many ways to get access tokens, but by far the simplest is Azure CLI. I showed this trick and many other Azure CLI tricks in a previous CODE Magazine article at https://codemag.com/Article/2001021/Azure-CLI.

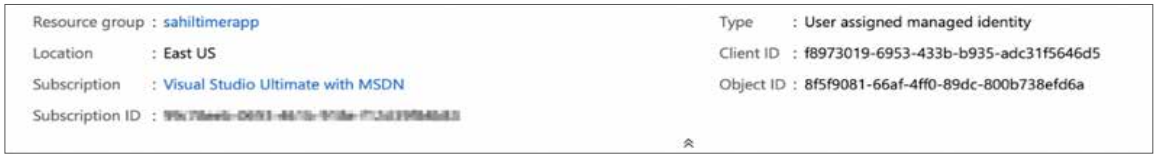


Figure 4: Newly created managed identity

On a computer with Azure CLI installed, first log in as an administrator to your Azure AD and issue the following command:

```
az account
  get-access-token
  --resource https://graph.microsoft.com
 | jq .\accessToken
 | pbcopy
```

This grabs an access token for you, with the necessary permissions, targeted for Microsoft Graph, and puts it in your clipboard.

With this token, I can grant permissions to my managed identity. This is simply a CURL command that issues a POST to Microsoft Graph, as shown in **Listing 1**.

Easy right? Uhm, not so much! There are so many magic GUIDs in **Listing 1** that I can't possibly move on without explaining each one of them. Let's slice and dice the command you see in **Listing 1**, bit by bit.

> Note that I am targeting a beta API, which is subject to change as it moves to a v1 API.

You're making a call to Microsoft Graph's service principal and making the necessary appRoleAssignments. 8f5f9081–66af-4ff0–89dc-800b738efd6a is the ObjectID of your managed identity service principal. 9654473a-512a-4c6a-8525–02cc112c5b08 is the GUID for Graph. And df021288-bdef-4463–88db-98f22de89214 is the GUID that represents User.Read.All.

Oh my! But how do you remember all those GUIDs? I don't! I look them up. Here is how.

The ObjectID of the managed identity is from **Figure 4**.

Getting the GUID for the service principal associated with Microsoft Graph is a bit more complex, but not too bad. And you can use these steps for any API. First, register an app in Azure AD and ensure that it has some kind of access to Microsoft Graph. Then, visit that app's service principal, and under there look for the **Permissions** menu item in the blade. There, choose Microsoft Graph, and you should see the service principal ID for Microsoft Graph. This can be seen in **Figure 7**.

Finally, to find the GUID that represents the application permission User.Read.All, grant this admin permission to any app, and open its manifest.xml. You should be able to grab that GUID from the requiredResourceAccess section.

Once you execute the command in **Listing 1**, you should get an output as shown in **Figure 8**.

You can verify the permissions by issuing an authenticated GET request to the same URL you sent a POST to.

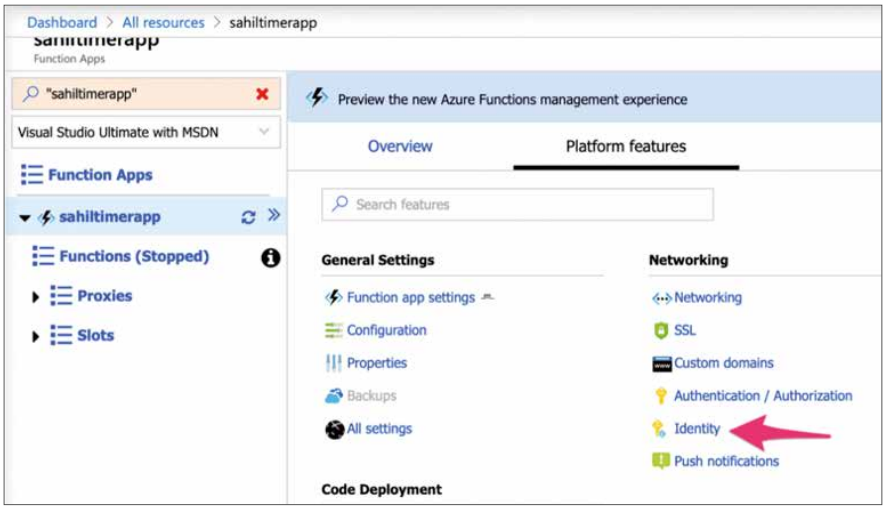


Figure 5: The identity section of a function app

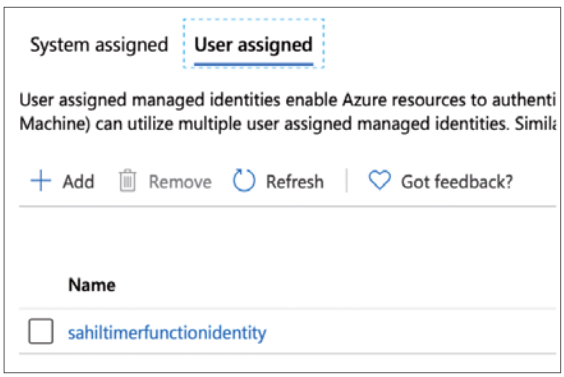


Figure 6: Assign the managed identity to the function app

Listing 1: Grant permissions to my managed identity

```
curl --location --request POST
,https://graph.microsoft.com/beta
 /servicePrincipals/
 9654473a-512a-4c6a-8525-02cc112c5b08
 /appRoleAssignments‘ \
--header ‚Authorization: Bearer <token_you_got_from_above>
‚ \
--header ‚Content-Type: application/json‘ \
--data-raw ‚{
 „principalId“: „8f5f9081-66af-4ff0-89dc-800b738efd6a“,
 „resourceId“: „9654473a-512a-4c6a-8525-02cc112c5b08“,
 „appRoleId“: „df021288-bdef-4463-88db-98f22de89214“
}‘
```

Alternatively, you can verify the permissions by going to enterprise applications and searching for your managed identity using its client ID, as shown in **Figure 9**. Note that you can grab the client ID of this managed identity from **Figure 4**.

Once you open the app that represents the managed identity, you can look for its permissions, as shown in **Figure 10**.

One important note: Just like any service principal or headless process that has no user identity, managed identity permissions must be applications permissions and granted ahead of time.

Author the Function App

All the plumbing is in place and all that's left to do is to author the function app. This part is perhaps the easiest. If you've installed the Azure function CLI on your computer, just type "func new" on terminal and it'll guide you through. Ensure that you choose to create a function app that accepts a timer trigger. The code for my function app using a timer trigger, can be seen in **Listing 2**.

This line from **Listing 2** is especially interesting!

```
const managedIdentityAppID =
  "f8973019-6953-433b-b935-adc31f5646d5";
```

That's the AppID of my user-assigned managed identity.

For this code to work, I've had to take a dependency on a specific node package. My package.json looks like this:

```
{
    "dependencies": {
        "request": "2.88.2",
        "@azure/identity": "^1.1.0"
    }
}
```

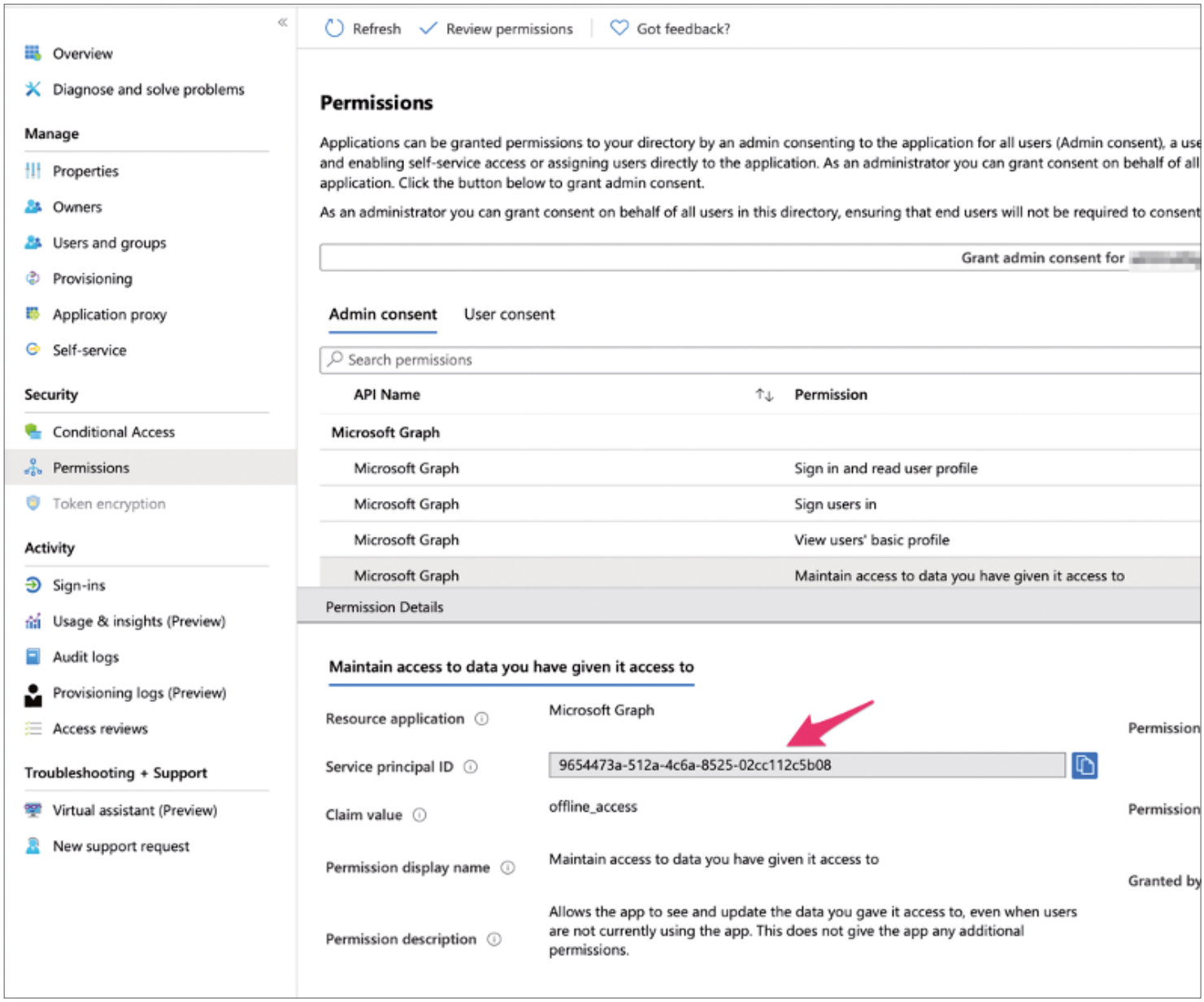


Figure 7: The service principal ID of Microsoft Graph

```
{
    "@odata.context": "https://graph.microsoft.com/beta/$metadata#appRoleAssignments/$entity",
    "id": "[redacted]",
    "creationTimestamp": "[redacted]",
    "appRoleId": "df021288-bdef-4463-88db-98f22de89214",
    "principalDisplayName": "sahiltimerfunctionidentity",
    "principalId": "8f5f9081-66af-4ff0-89dc-800b738efd6a",
    "principalType": "ServicePrincipal",
    "resourceDisplayName": "Microsoft Graph",
    "resourceId": "9654473a-512a-4c6a-8525-02cc112c5b08"
}
```


Figure 8: Assign permission success

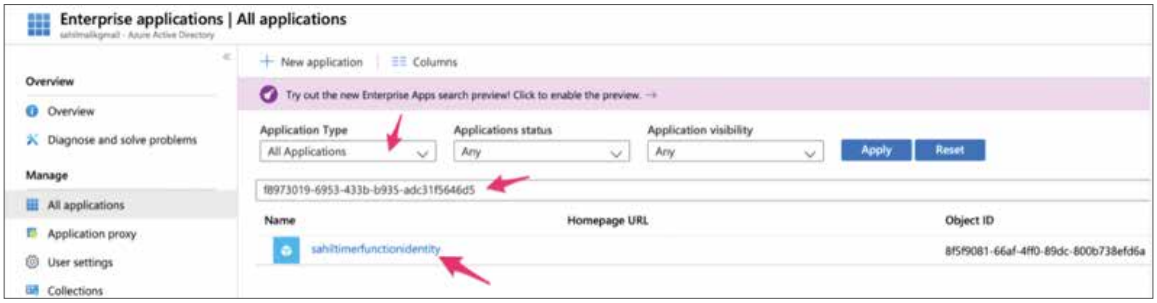


Figure 9: Finding your managed identity.

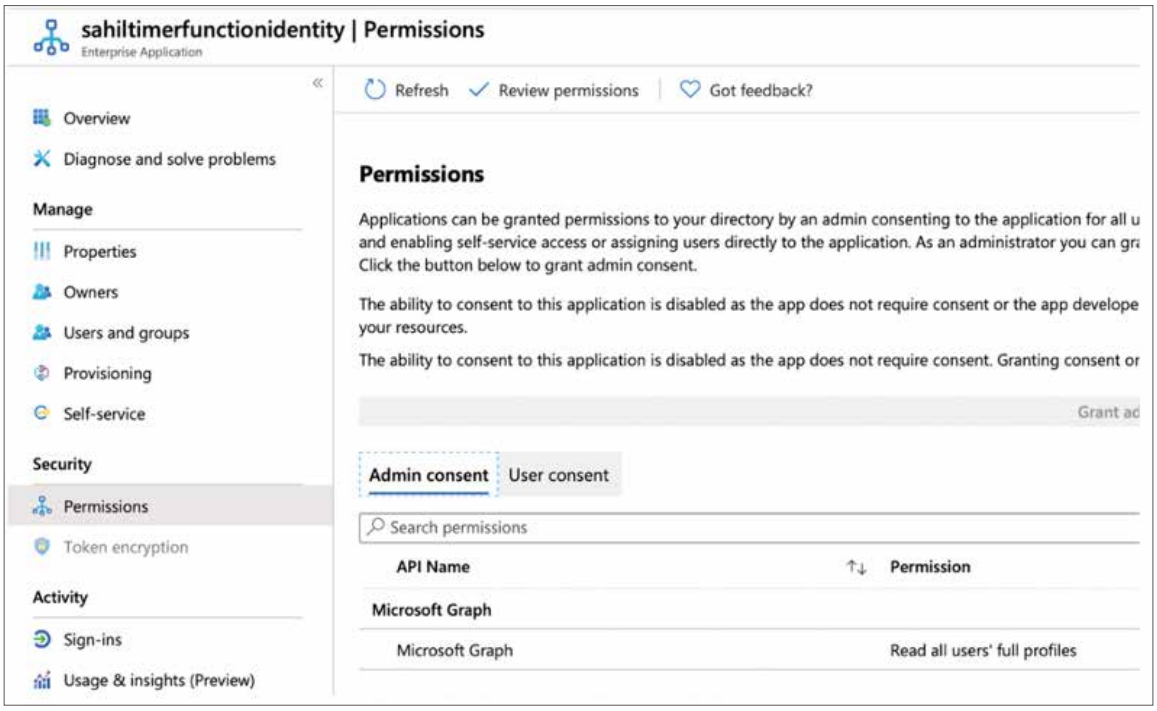


Figure 10: The managed identity permission

Running the Code Example

At this point, all my code changes are done. Now simply go ahead and deploy the function app and connect to it streaming logs. You should see an output like that in **Figure 11**.

As can be seen from **Figure 11**, you're able to get an access token for the managed identity to come up with the necessary permissions and were able to call Microsoft Graph with it.

This is incredibly powerful. You have just authored a timer function, which is a headless process for which you'd never need to manage any credentials. No more key rotation headache. No more headache of securing that credential. Everything is contained neatly inside Azure.

Listing 2: The function app code

```
const identity = require('@azure/identity');
const request = require('request');

module.exports = async function (context, myTimer) {
    var timeStamp = new Date().toISOString();
    context.log(
       'Timer function called', timeStamp);
    var promise =
      new Promise((resolve, reject) => {
        const managedIdentityAppID =
         "f8973019-6953-433b-b935-adc31f5646d5";
        const credential =
            new
   identity.ManagedIdentityCredential(
    managedIdentityAppID);
        context.log('getting access token');
        credential.getToken(
          "https://graph.microsoft.com/.default")
            .then(response => {
                context.log(response);
    request.get(
   'https://graph.microsoft.com/v1.0/users',
{
 'auth': {
'bearer': response.token}
      },
      function (error, response, body) {
       if (error) {
        reject(error);
       }
       context.log(timeStamp + body);
         resolve(body);
       })}, err => {
          context.log(err);
          reject(err);
        });
    });
    await promise;
};
```

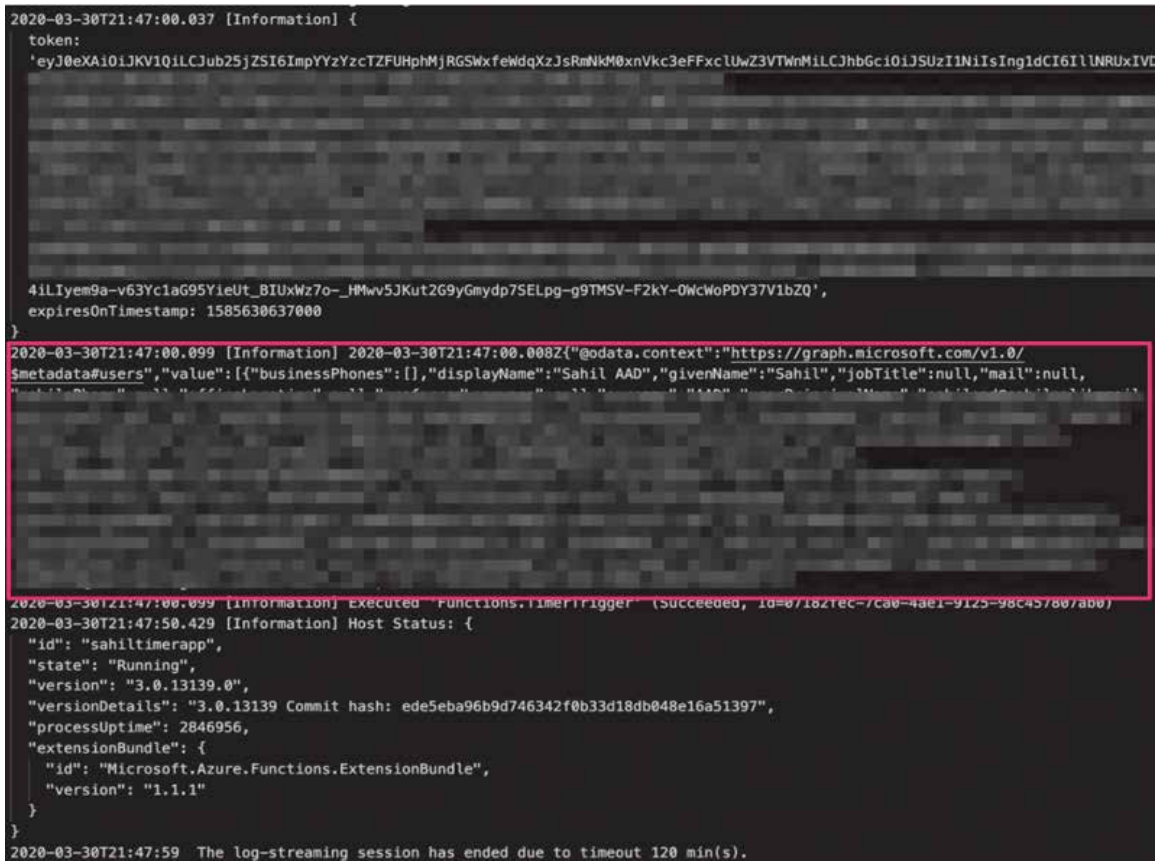


Figure 11: Our call as a managed identity successfully calling Microsoft Graph

Summary

A platform such as Azure is full of golden nuggets everywhere. I've been working with Azure for many years now and yet I feel I've only scratched the surface of it. Every day I learn so many more new and interesting ways of solving customers' problems.

Managed identity is a very powerful feature of Azure. It allows you to write secure applications because you have no credentials to manage. There's no danger of polluting environment variables. There's no danger of accidentally checking in credentials into source code for the world to see. It promotes better architecture. It promotes better patterns. It reduces stupid mistakes.

Although this article showed you one possible usage of managed identity, using it as headless process to call protected APIs, the possibilities are truly endless. I'm sure you'll use your own ingenuity and creativity to come up with some amazing examples of managed identity put to real use.

I look forward to seeing them. Stay in touch and happy coding.

Sahil Malik

INDUSTRY'S FASTEST DATA CONNECTORS

Next-Generation Data Connectivity for BI, ETL & Analytics

Modern business depends on real-time data analysis to deliver actionable insights and drive growth. Our drivers offer the fastest and easiest way to connect BI, Analytics, ETL, and custom applications to that data, wherever it resides.

Our data connectivity tools support a new era of analytics and cognitive computing. Modern users demand that applications retrieve and process massive datasets at unprecedented speeds, and our drivers have been specifically engineered and optimized for data-intensive reporting and analysis.

Data Connectivity Features:

- 200+ SaaS, NoSQL, & Big Data Connectors
- Real-Time Bi-directional Data Access
- Rich Metadata Discovery
- Robust SQL-92 Support
- High-Performance Data Connectivity

www.cdata.com

ODBC JDBC ADO ODATA SQL SSIS BIZTALK POWER BI EXCEL MOBILE

Seamless Data Integration

All drivers expose rich metadata, critical for automated data exploration and discovery. Includes intelligent row-scan, type detection, relationship exploration and support for unstructured data.

Robust SQL-92 Engine

Built on a rich SQL-92 engine with support for bulk operations, push-down, client functions, and aggregation, our drivers seamlessly handle even the most complex queries from modern BI & Reporting.

Unmatched Performance

With optimized performance down to the socket level, our drivers offer unmatched performance. Read and write speeds over twice as fast as other SaaS, NoSQL & Big Data connectivity solutions.

Use the MVVM Design Pattern in MVC Core: Part 2

In Part 1 of this article series (called "Use the MVVM Design Pattern in MVC Core: Part 1," and located at https://bit.ly/3gA4IkL), you started using the Model-View-View-Model (MVVM) design pattern in MVC Core applications. In that article, you created an MVC Core application using VS Code and a few class library projects in which to put your entity, repository, and view model classes.

Paul D. Sheriff
http://www.pdsa.com

Paul has been in the IT industry over 33 years. In that time, he has successfully assisted hundreds of companies to architect software applications to solve their toughest business problems. Paul has been a teacher and mentor through various mediums such as video courses, blogs, articles, and speaking engagements at user groups and conferences around the world.
Paul has 23 courses in the www.pluralsight.com library (http://www.pluralsight.com/author/paul-sheriff) on topics ranging from JavaScript, Angular, MVC, WPF, XML, jQuery, and Bootstrap. Contact Paul at psheriff@pdsa.com.

Using the AdventureWorksLT database, you created a page to display product data in an HTML table. In addition, you wrote the appropriate code to search for products based on user input.

In this article, you're going to add on to the sample from the last article to sort the database when the user clicks on any of the column headers in the HTML table. You're going to learn how to add a pager to your HTML table so only a specified number of rows are displayed on the page. Finally, you learn to cache the product data in the Session object to improve performance.

Sort the Product Table

In most Web applications, when you have an HTML table of data, the user can click on the header above each column and sort the data within that column. If the user clicks on the same column header twice in a row, it first sorts the data in ascending, then in descending order. If the user clicks on a different column header, the sort direction should go back to ascending and the table should be sorted on the new column's data.

Add Sort Properties to ViewModel Base Class

The ViewModelBase class you created in the previous article is designed to be the base class for any view models you add to your project. As many pages you design might need the sorting functionality, add three new properties to the ViewModelBase class, as shown in the following code snippet:

```
public string SortDirection { get; set; }
public string SortExpression { get; set; }
public string PreviousSortExpression
  { get; set; }
```

Initialize the three properties you just added by adding code into the constructor, as shown below.

```
public ViewModelBase() {
  EventCommand = string.Empty;
  SortDirection = "asc";
  SortExpression = string.Empty;
  PreviousSortExpression = string.Empty;
}
```

You put either "asc" or "desc" to specify the sort direction into the *SortDirection* property. The *SortExpression* property holds the name of the column the user just clicked upon. The *PreviousSortExpression* property holds the name of the last column the user clicked upon. If the *SortExpression* property is equal to the *PreviousSortExpression*, change the *SortDirection* to the "desc" if its current value is "asc", or to "asc" if its value is "desc". After you change the *SortDirection*, put the *SortExpression* value into the *PreviousSortExpression* property. Open the **ViewModelBase.cs** file and add a new method named SetSortDirection() to perform this logic.

```
protected virtual void SetSortDirection() {
  if (SortExpression ==
      PreviousSortExpression) {
    // Toggle the sort direction if
    // the field name is the same
    SortDirection = (SortDirection == "asc" ?
       "desc" : "asc");
  }
  else {
    SortDirection = "asc";
  }

  // Set previous sort expression to new column
  PreviousSortExpression = SortExpression;
}
```

Create Hidden Input Fields for New Properties

In the last article, you added a partial page with hidden input fields to hold the values of each property added to the ViewModelBase class. The *EventCommand* property is the only hidden input field in this partial page at this point. Open the **_StandardViewModelHidden.cshtml** file and add three more hidden input fields to hold the values of each of the three new properties you added to the ViewModelBase class. Modify the partial page to look like the following code snippet:

```
@model MVVMViewModelLayer.ViewModelBase

<input type="hidden" asp-for="EventCommand" />
<input type="hidden" asp-for="SortDirection" />
<input type="hidden" asp-for="SortExpression" />
<input type="hidden"
  asp-for="PreviousSortExpression" />
```

Create Clickable Column Headers

You need to change each column header to hyperlinks so the user can click on them. To do this, use an anchor tag with a couple of "data-" attributes added to pass data to the view model. Add an attribute named "data-custom-cmd" and set its value equal to "sort". This tells the view model what operation to perform. Add another attribute named "data-custom-arg" and set its value to the property name to sort upon. This value will be different for each column the user clicks on. Open the partial page named **_ProductList.cshtml** and add the links to each of the column headers, as shown in **Listing 1**.

Set the SortExpression Property Using jQuery

In the last article, you added a $(document).ready() function to connect up a click event to any HTML element that

has a "data-custom-cmd" attribute. Within that code, you retrieve the value in the "data-custom-cmd" attribute and put it into the hidden input field that is bound to the *EventCommand* property in your view model. If the *EventCommand* property is set to "sort", retrieve the value in the "data-custom-arg" attribute and put that value into the hidden input field bound to the *SortExpression* property. Open the **Products.cshtml** and locate the $(document).ready() function at the bottom of the file. After the line of code that updates the $("#EventCommand") hidden input field add an If statement and the code within it as shown below.

```
// $(document).ready()

// Fill in command to post back to view model
$("#EventCommand").val($(this)
  .data("custom-cmd"));

// Only set sort variables if command was "sort"
if($("#EventCommand").val() == "sort") {
  // Get the new sort expression
  $("#SortExpression").val(
   $(this).data("custom-arg"));
}
```

Add SortProducts Method

Now that you have the *EventCommand* and *SortExpression* properties filled in with the proper values, and you have code written in the SetSortDirection() method to set the *SortDirection* and *PreviousSortExpression* properties, it's time to do the actual sorting. Open the **ProductViewModel.cs** file and add a SortProducts() method, as shown in **Listing 2**.

The first thing the SortProducts() method does is to retrieve the product data from the data repository by calling the SearchProducts() method. Once the *Products* collection has been filled in with the product data, call the SetSortDirection() method. Check the *SortDirection* property to see if it is set to "asc" or "desc". Set a Boolean variable appropriately. In the switch() expression, compare the value in the *SortExpression* property against each case statement. Once you have a match on the column name, check the Boolean variable to see if you should apply the OrderBy() or OrderByDescending() methods to the *Products* collection.

Add Sort Command to HandleRequest() Method

The HandleRequest() method is the public method called from the controller. This method is where you check the *EventCommand* property to decide what method(s) to call in the view model to set the view model properties for displaying on the screen. You just passed in a new command with a value of "sort", so you need to add a new case statement just above the "search" case. Modify the method called from SearchProducts() to SortProducts().

Listing 1: Change the table headers to hyperlinks for sorting

```
<th>
  <a href="#"
      data-custom-cmd="sort"
      data-custom-arg="Name">
      Product Name
  </a>
</th>
<th>
  <a href="#"
      data-custom-cmd="sort"
      data-custom-arg="ProductNumber">
      Product Number
  </a>
</th>
<th class="text-right">
  <a href="#"
      data-custom-cmd="sort"
      data-custom-arg="StandardCost">
      Cost
  </a>
</th>
<th class="text-right">
  <a href="#"
      data-custom-cmd="sort"
      data-custom-arg="ListPrice">
      Price
  </a>
</th>
```

Listing 2: For each column in your table you need code to sort data based on that field.

```
protected virtual void SortProducts() {
  // Search for Products
  SearchProducts();

  if (EventCommand == "sort") {
    // Set sort direction
    SetSortDirection();
  }

  // Determine sort direction
  bool isAscending = SortDirection == "asc";

  // What field should we sort on?
  switch (SortExpression.ToLower()) {
    case "name":
      if (isAscending) {
        Products = Products.OrderBy(
          p => p.Name).ToList();
      } else {
        Products = Products.OrderByDescending(
          p => p.Name).ToList();
      }
      break;
    case "productnumber":
      if (isAscending) {
        Products = Products.OrderBy(
          p => p.ProductNumber).ToList();
      } else {
        Products = Products.OrderByDescending(
          p => p.ProductNumber).ToList();
      }
      break;
    case "standardcost":
      if (isAscending) {
        Products = Products.OrderBy(
          p => p.StandardCost).ToList();
      } else {
        Products = Products.OrderByDescending(
          p => p.StandardCost).ToList();
      }
      break;
    case "listprice":
      if (isAscending) {
        Products = Products.OrderBy(
          p => p.ListPrice).ToList();
      } else {
        Products = Products.OrderByDescending(
          p => p.ListPrice).ToList();
      }
      break;
  }
}
```

```
case "sort":
case "search":
  SortProducts();
  break;
```

Set Sort Order in Controller

When the user first enters your product list page, show them the products sorted by the *Name* property. Open the **ProductsController.cs** file and add two lines of code to the Products() method, as shown in the code snippet below.

```
public IActionResult Products()
{
  // Load products
  _viewModel.SortExpression = "Name";
  _viewModel.EventCommand = "sort";
  _viewModel.HandleRequest();

  return View(_viewModel);
}
```

The first line of code sets the *SortExpression* property to "Name". The second line of code sets the *EventCommand* property to "sort". When the HandleRequest() method is called, the SortProducts() method is called because of the value in the *EventCommand* property. Because the value in the *PreviousSortExpression* property is blank and the value in the *SortExpression* property is "Name", the two values are not equal. This causes the *SortDirection* property to be set to "asc". Because of the *SortExpression* and *SortDirection*, the switch statement causes the *Products* collection to be ordered by the *Name* property in an ascending order.

Reset Hidden Fields After Postback

When the user clicks on another column header and forces a post-back with new values to sort by, the [HttpPost] Products() method is called. After the HandleRequest() method is called, there are new values in the *SortExpression*, *PreviousSortExpression* and *SortDirection* properties. To update the hidden input fields, call the ModelState.Clear() method as shown in the code snippet below.

```
[HttpPost]
public IActionResult Products(
  ProductViewModel vm)
{
  vm.Repository = _repo;
  vm.HandleRequest();

  ModelState.Clear();

  return View(vm);
}
```

Try It Out

Now that you have code written to perform sorting, it is time to try it out. Run the Web application and click on the different column headers. Try clicking on different headers to see the data sort in ascending order for each column. Then try clicking on the same column header to see the data swap between ascending and descending order for that column of data.

Create a Common Library Project

The next step is to add paging to your table. In order to do this, you need some classes to help with the paging. These classes are in the download for this article under a folder called PagerClasses. Please download the samples now so you can continue following along.

These pager classes you downloaded belong in another class library that you are naming CommonLibrary. This class library will be for any generic classes that can be used in any type of application. To create the CommonLibrary project, open a terminal window by clicking on the **Terminal > New Terminal** menu. Go to your development directory (in my case, that was D:\Samples). Type in the following commands to create a new folder named CommonLibrary.

```
MD CommonLibrary
CD CommonLibrary

dotnet new classlib
```

The dotnet new classlib command creates a class library project with the minimum number of references to .NET Core libraries. Now that you have created this new project, add it to your Visual Studio Code workspace by clicking the **File > Add Folder to Workspace...** menu. Select the CommonLibrary folder and click on the **Add** button. You should see the CommonLibrary project added to your VS Code workspace. Delete the **Class1.cs** file, as you're not going to need that. Copy the \PagerClasses folder into the CommonLibrary folder so it now appears in your VS Code workspace.

Add References to the Common Library

This CommonLibrary needs to be referenced from both the MVVMViewModelLayer and the MVVMSample projects. Click on the **Terminal > New Terminal** menu and select the MVVMSample folder. Set a reference to the CommonLibrary project using the following command.

```
dotnet add . reference
  ../CommonLibrary/CommonLibrary.csproj
```

Change the directory to the MVVMViewModelLayer folder and execute the following command to reference the CommonLibrary in this project as well.

```
dotnet add . reference
  ../CommonLibrary/CommonLibrary.csproj
```

Try It Out

To ensure that you've typed in everything correctly, run a build task to compile the projects. Because you have a reference from the MVVMSample to the CommonLibrary, if you run a build task on the MVVMSample project, it builds all of the other projects, including the CommonLibrary. Select the **Terminal > Run Build Task...** menu and select the MVVMSample project. Watch the output in the terminal window and you should see that it compiles all five of your projects.

Paging

Instead of displaying hundreds of product rows in a table and forcing the user to scroll down on your Web page, add a pager to your table like the one shown at the bottom of **Figure 1**. There are two pieces to adding a pager to your table: the first is the actual pager UI, and the second is the calculation and selection of which rows are on each page. All the code to do the calculations and create a set of pager items used to display the pager are contained in the classes in the

Getting the Sample Code

You can download the sample code for this article by visiting www.CODEMag.com under the issue and article, or by visiting www.pdsa.com/downloads. Select "Fairway/PDSA Articles" from the Category drop-down. Then select "Use the MVVM Design Pattern in MVC Core - Part 2" from the Item drop-down.

\PageClasses folder you just added to the CommonLibrary project. I'm not going to cover how these classes work, as that is beyond the scope of this article. I am going to show you how to use them to create the pager shown in **Figure 1**.

To create a pager that looks like what you see in **Figure 1**, use Bootstrap and the HTML shown in **Listing 3**. Add the "data-custom-cmd" and "data-custom-arg" attributes to each anchor tag to set properties in the view model class. The "data-custom-arg" attribute is going to be used to set a property named *EventArgument* in the ViewModelBase class. In the previous section on sorting, you used the "data-custom-arg" attribute to set the *SortExpression* property. For paging, you are going to set the *EventArgument* property from the "data-custom-arg" attribute and use that to tell the pager how to page through the data.

Add Paging Properties to View Model Base Class

Open the **ViewModelBase.cs** file and add a few new properties to help with paging. The first one, *EventArgument*, receives the action to perform when paging. For example, this can be set to "next", "previous", "first", "last", or a specific page number, as you can see by looking at the "data-custom-arg" attributes in **Listing 3**. The next property, *Pager*, is of the type Pager, which is one of the classes you downloaded and added to the CommonLibrary project from the PagerClasses folder. This class contains properties such as *PageSize*, *VisiblePagesToDisplay*, *PageIndex*, *StartingRow*, *TotalPages*, and *TotalRecords*. The last property to add to the view model, *Pages*, is a collection of PagerItem objects. A *PagerItem* object represents a single visible anchor tag in the pager displayed at the bottom of the table. Each *PagerItem* object contains properties such as *Text*, *Tooltip*, *Argument*, and *CssClass*.

```
public string EventArgument { get; set; }
public Pager Pager { get; set; }
public PagerItemCollection Pages { get; set; }
```

After adding these new properties, initialize the *EventArgument* and *Pager* properties in the constructor of the ViewModelBase class using the code shown in the following code snippet:

Product Name	Product Number	Cost	Price
All-Purpose Bike Stand	ST-1401	$59.47	$159.00
AWC Logo Cap	CA-1098	$6.92	$8.99
Bike Wash - Dissolver	CL-9009	$2.97	$7.95
Cable Lock	LO-C100	$10.31	$25.00
Chain	CH-0234	$8.99	$20.24

« ‹ 1 2 3 4 5 6 7 8 9 10 ... › »

Figure 1: Add a pager below your table to keep the number of records displayed at one time to a minimum.

```
public ViewModelBase() {
  EventCommand = string.Empty;
  EventArgument = string.Empty;
  Pager = new Pager();
  SortDirection = "asc";
  SortExpression = string.Empty;
  PreviousSortExpression = string.Empty;
}
```

Add a new method named SetPagerObject() to the ViewModelBase class. This method accepts the total amount of records in the Product table. This parameter is passed to the *Pager.TotalRecords* property. The *TotalPages* property in the *Pager* object is calculated from the *TotalRecords* and the *PageSize* properties. *PageSize* has a default size of ten. If the *TotalRecords* is set to 50, the total pager objects to show on the UI is five.

Listing 3: Create a pager using HTML and Bootstrap

```
<ul class="pagination">
  <li class="page-item disabled">
    <a class="page-link" href="#"
        data-custom-cmd="page"
        data-custom-arg="first">
        &laquo;
    </a>
  </li>
  <li class="page-item disabled">
    <a class="page-link" href="#"
        data-custom-cmd="page"
        data-custom-arg="prev">
        &lsaquo;
    </a>
  </li>
  <li class="page-item active">
    <a class="page-link" href="#"
        data-custom-cmd="page"
        data-custom-arg="0">
        1
    </a>
  </li>
  <li class="page-item ">
    <a class="page-link" href="#"
        data-custom-cmd="page"
        data-custom-arg="1">
        2
    </a>
  </li>
<li class="page-item ">
    <a class="page-link" href="#"
        data-custom-cmd="page"
        data-custom-arg="10">
        ...
    </a>
  </li>
  <li class="page-item ">
    <a class="page-link" href="#"
        data-custom-cmd="page"
        data-custom-arg="next">
        &rsaquo;
    </a>
  </li>
  <li class="page-item ">
    <a class="page-link" href="#"
        data-custom-cmd="page"
        data-custom-arg="last">
        &raquo;
    </a>
  </li>
</ul>
```

```
protected virtual void
  SetPagerObject(int totalRecords) {
  // Set Pager Information
  Pager.TotalRecords = totalRecords;

  // Set Pager Properties
  Pager.SetPagerProperties(EventArgument);

  // Build paging collection
  Pages = new PagerItemCollection(Pager);
}
```

Next, the SetPagerProperties() method is called and passed the *EventArgument* property. Remember the *EventArgument* is set to "next", "previous", or similar commands depending on which anchor tag the user clicked upon. From this command, the *PageIndex* property in the *Pager* object is set.

Now that you have the properties in the *Pager* object calculated, pass this *Pager* object to the *PagerItemCollection* object. By reading the *TotalRecords*, the *TotalPages* and the *PageIndex* properties in the *Pager* object, the *PagerItemCollection* builds the collection of pages used to display the anchor tags.

Create Pager UI on the Product List Page

To build the pager that the user sees on the list page, open the **_ProductList.cshtml** file and add a using statement at the top of the file to reference the CommonLibrary.PagerClasses namespace. It's in this namespace that the Pager classes are located.

```
@using CommonLibrary.PagerClasses
```

Move to the bottom of this file and add the code in this next snippet. This code is what takes the *Pages* property you just created in the SetPagerObject() method and builds each hyperlink you see in the pager at the bottom of **Figure 1**. Feel free to step through the code in the *PagerItemCollection* object to see how this pager is built.

```
<ul class="pagination">
  @foreach (PagerItem item in Model.Pages) {
    <li class="page-item @item.CssClass">
      <a class="page-link" href="#"
          data-custom-cmd="page"
          data-custom-arg="@item.Argument"
          title="@item.Tooltip">
          @Html.Raw(item.Text)
      </a>
    </li>
  }
</ul>
```

Get One Page of Data from the Products Collection

Before you can try out the paging, you have a few more pieces of code to write. First, open the **ProductViewModel.cs** file and add a using statement at the top of the file.

```
using CommonLibrary.PagerClasses;
```

Within the ProductViewModel class, add a new property to hold onto the total amount of product records you read from the table. The *TotalProducts* property is used to display how many records the user selected and display it on the page.

```
public int TotalProducts { get; set; }
```

Add a new method named PageProducts() that sets the *TotalProducts* property, then calls the SetPagerObject() method in the base class. Once the *Pager* object has been set up, you now need to get just the data for the current page of data. The starting row in the *Products* collection can be calculated by taking the current *PageIndex* property and multiplying that number by the *PageSize* property. Pass this result to the LINQ Skip() method on the *Products* collection. Then take the next *PageSize* number of records from the collection.

```
protected virtual void PageProducts() {
  TotalProducts = Products.Count;

  base.SetPagerObject(TotalProducts);

  Products = Products.Skip(
       base.Pager.PageIndex *
       base.Pager.PageSize)
     .Take(base.Pager.PageSize).ToList();
}
```

Locate the HandleRequest() method and modify the switch statement to handle the different commands. Modify the "search" command to reset the *PageIndex* property. Add a "page" command for when the user clicks on one of the pag-

er anchor tags. Call the PageProducts() method after the call to the SortProducts() method in both the "search" and "page" case statements.

```
case "search":
  Pager.PageIndex = 0;
  SortProducts();
  PageProducts();
  break;
case "sort":
case "page"
  SortProducts();
  PageProducts();
  break;
```

Add Hidden Input Fields for Paging

Open the **_StandardViewModelHidden.cshtml** file and add two new hidden input fields. You need to post-back both the *EventArgument* and the current *PageIndex* values. The *EventArgument* is going to have the "data-custom-arg" value from the pager and the *PageIndex* is the page the user is currently viewing.

```
<input type="hidden" asp-for="EventArgument" />
<input type="hidden"
  asp-for="Pager.PageIndex" />
```

Add Total Records to Products Search Page

You added the *TotalProducts* property to the ProductViewModel class. It's now time to display that on the page. Put the following code within the card-footer in the **_ProductSearch.cshtml** file.

```
<div class="card-footer bg-primary text-light">
  <div class="row">
    <div class="col-8">
      <button type="button"
              data-custom-cmd="search"
              class="btn btn-success">Search
      </button>
    </div>
    <div class="col-4">
      <p class="text-right">
        Total Records: @Model.TotalProducts
      </p>
    </div>
  </div>
</div>
```

Set the EventArgument Property Using jQuery

The last thing you need to do before trying out the paging is to take the "data-custom-arg" attribute value and place it into the new hidden input field you just added. Open the **Products.cshtml** file and locate the $(document).ready() function and after the line of code that updates the $("#EventCommand") hidden input field, add a new line of code to set the *EventArgument* as shown in the code snippet below:

```
// The $(document).ready() function

// Fill in command to post back to view model
$("#EventCommand").val($(this)
  .data("custom-cmd"));

// Fill in arguments to post back to view model
$("#EventArgument").val($(this)
  .data("custom-arg"));
```

Try It Out

You're now ready to try out the paging to ensure that you typed everything in correctly. Run the application and try clicking on the various anchor tags on the pager UI.

Changing the Page Size

By default, the *PageSize* property in the Pager object is set to 10. If you wish to adjust the size, modify the *PageSize* property in the controller. Open the **ProductsController.cs** file and, in the Products() method, add the following line of code just before the call to the HandleRequest() method.

```
_viewModel.Pager.PageSize = 5;
```

Also, in the [HttpPost] Products() method, add the following line of code just before the call to the HandleRequest() method.

```
vm.Pager.PageSize = 5;
```

Try It Out

Run the application again and you should see only five rows of data appear on each page in the table.

Cache the Products List

As you may have noticed, the Products are read from the database each time you sort or page through the data. Having to retrieve the data every time from the data is very inefficient and this process can be fixed easily. If your product data does not change that often, cache the data in your Web server's memory. There are many methods you may employ for caching; for this article, let's use the Session object.

Add Property to Hold all the Product Data

In the Product view model class, there is a *Products* collection to hold the data for the one page of data to display. To avoid a round-trip to your SQL Server to get all your product data, add another property to hold the original list of product data. The list of all product data is placed into your cache after retrieving the data the first time. The data is retrieved from the cache on each post-back to populate the *Products* collection when the user requests the next page of data to display. Open the **ProductViewModel.cs** file and add a new property named *AllProducts*.

```
public List<Product> AllProducts { get; set; }
```

Each time the user posts back to the controller to either search, sort, or page, the SearchProducts() method is called to retrieve the data from the Repository class. In this method, modify the code to use the data in the *AllProducts* collection if that collection has been retrieved from the cache.

Locate the SearchProducts() method and modify it to look like **Listing 4**. In SearchProducts(), check if this is the first time the user has hit the product list page. If the *AllProducts* collection is null, then call the Search() method on the Repository class to get all of the product data from the database server and store that data into the *AllProducts* property.

In the Products() method in the ProductsController class, you're going to store the *AllProducts* collection into the

Listing 4: Modify the SearchProducts() method to use the cached data

```
public virtual void SearchProducts() {
  if (AllProducts == null) {
    if (Repository == null) {
      throw new ApplicationException(
        "Must set the Repository property.");
    } else {
      // Get data from Repository
      AllProducts = Repository
        .Search(SearchEntity)
        .OrderBy(p => p.Name).ToList();
    }
  }
  // Get data for the Products collection
  Products = AllProducts.Where(p =>
    (SearchEntity.Name == null ||
      p.Name.StartsWith(SearchEntity.Name)) &&
      p.ListPrice >= SearchEntity.ListPrice)
        .ToList();
}
```


Session object. You're then going to retrieve the data from the Session object in the [HttpPost] Products() method and put that data back into the *AllProducts* collection. When the SearchProducts() method is called the second time, the *AllProducts* collection is not null and thus the *Products* collection is built by querying the *AllProducts* collection instead of going to the database server.

Add Session State to MVC Core

As mentioned, you are going to use the Session state object that's built-in to MVC Core. In order to use Session, you must configure it in the Startup class. Open the **Startup.cs** file and locate the InjectAppServices() method you created in the last article. After the line of code services.AddDbContext(), add the following two lines of code to add services to support session state.

```
// The next two lines are for session state
services.AddDistributedMemoryCache();
services.AddSession();
```

Next, locate the Configure() method and find the line of code app.UseRouting(). After this line of code, add the following line of code to turn on session state.

```
app.UseSession();
```

Add Newtonsoft Package

The Session object in MVC Core can only store strings or integers. However, you need to put in a collection of Product objects. In order to do this, you need to serialize the collection into JSON. This can be done using the Newtonsoft package. Open a terminal window in the MVVMSample folder and execute the following command to bring in the Newtonsoft library.

```
dotnet add package Newtonsoft.Json
```

Store AllProducts Collection into Session

It's now time to cache the data you placed into the *AllProducts* collection. Open the **ProductsController.cs** file and add some Using statements to the top of the file.

```
using System.Collections.Generic;
using Microsoft.AspNetCore.Http;
using MVVMEntityLayer;
using NewtonSoft.Json;
```

Locate the Products() method and after the call to HandleRequest() has been called, get the data from the *AllProducts* collection, serialize it, and place it into the Session object using the SetString() method as shown in the code below.

```
HttpContext.Session.SetString("Products",
  JsonConvert.SerializeObject(
    _viewModel.AllProducts));
```

When the user clicks on any of the links that post back to the controller, the [HttpPost] Products() method is called. Place the following line of code just before the call to the HandleRequest() method to populate the *AllProducts* collection with the data you stored into the Session object.

```
vm.AllProducts =
  JsonConvert.DeserializeObject<List<Product>>
    (HttpContext.Session.GetString("Products"));
```

When the HandleRequest() method is called, the *AllProducts* collection has data in it, and thus in the SearchProducts() method, the Products collection is built from the data in this collection instead of calling the database server.

Try It Out

Run the application and try out the various features of the page. Try searching for data, try sorting the data, and try paging through the data. If you set a breakpoint, in the SearchProducts() method, you can see that it only fetches the data from the Repository class the first time into the page. After that, the data is always coming from the cached data.

Summary

In this article, you learned how to add hyperlinks to each column header in a table to allow sorting of product data. Next, you added paging capabilities to the table, so the user only sees a limited amount of data at a time. This is a better user experience than having a large scrollable table. Finally, you learned how to use the Session object in order to avoid a round-trip to the database server each time the user clicks on something on the page. This saves time and resources and makes updating your page much quicker. The great thing about using the MVVM design pattern is the code in your controller is still very simple, and your view model class need only expose one public method. In the next article, you'll learn to add, edit, and delete product data using the MVVM design pattern.

Paul D. Sheriff

CODE

CODE SECURITY

DREADING SHARING THAT YOUR APPLICATION CAUSED A DATA BREACH?

CODE SECURITY'S APPLICATION TESTING SERVICES CAN HELP YOU PREVENT THAT BREACH.

- My development team practices secure coding techniques ☐ Yes ☒ No
- My applications have been tested for security vulnerabilities ☐ Yes ☒ No
- Code Audits and Penetration Testing have been performed by a third party ☐ Yes ☒ No

CODE Security experts can help you identify and correct security vulnerabilities in your products, services and your application's architecture. Additionally, CODE Training's hands-on **Secure Coding for Developers** training courses educates developers on how to write secure and robust applications.

Contact us today for a free consultation and details about our services.

codemag.com/security
832-717-4445 ext. 9 • info@codemag.com

Transform Your ASP.NET Core API into AWS Lambda Functions

In a recent CODE Magazine article (Discovering AWS for .NET Developers, May/June 2020 https://www.codemag.com/Article/2005041), you read about my first foray into using .NET in the Amazon Web Services (AWS) ecosystem. I explored the AWS Relational Database Service (RDS) and created an ASP.NET Core API using Entity Framework Core (EF Core) to connect to a SQL Server Express database hosted in RDS. In the end, I deployed my API to run on AWS Elastic Beanstalk with my database credentials stored securely in Amazon's Parameter Store to continue interacting with that same database.

Julie Lerman
thedatafarm.com/blog
twitter.com/julielerman

Julie Lerman is a Microsoft Regional Director, Docker Captain and a long-time Microsoft MVP who now counts her years as a coder in decades. She makes her living as a coach and consultant to software teams around the world. You can find Julie presenting on Entity Framework, Domain Driven Design and other topics at user groups and conferences around the world. Julie blogs at thedatafarm.com/blog, is the author of the highly acclaimed "Programming Entity Framework" books, the MSDN Magazine Data Points column and popular videos on Pluralsight.com. Follow Julie on twitter at julielerman.

Interacting with the database was a great first step for me and hopefully for readers as well. And it gave me enough comfort with AWS to set my sights on their serverless offering, AWS Lambda Functions. Some of the most critical differences between hosting a full application in the cloud and rendering your logic as functions are:

- Rather than paying for an application that's constantly running and available, you only pay for individual requests to a function. In fact, the first one million requests each month are free along with a generous amount of compute time. (Details at aws.amazon.com/Lambda/pricing)
- Serverless functions are also stateless, meaning that you can run many instances of the same function without worrying about conflicting state across those instances.
- Most of the management of serverless functions is taken care of by the function host, leaving you to focus on the logic you care about.

In this article, I'll evolve the ASP.NET Core API from the previous article to a Serverless Application Model (SAM) application which is a form of Lambda function.

Moving an Existing ASP.NET Core API to a Serverless App

This was such an interesting journey. And an educational one. Amazon has created what I'll refer to as a lot of "shims" to seamlessly host an ASP.NET Core API behind a Lambda function. The beauty of this is that you can write an ASP.NET Core API using the skills you already have and AWS's logic will provide a bridge that runs each controller method as needed. That way, you get the benefits of serverless functions such as the on-demand billing but continue to build APIs the way you already know how. It took a bit of time (and some repeated explanations and reading) to wrap my head around this. I hope this article provides a quicker learning path for you.

If you installed the AWS Toolkit for Visual Studio as per the previous article, then you already have the project template needed to create the basis for the new API. I'll start by creating a new project using the template and then copy the classes and some code from the existing API into the new project. The project template contains part of the "bridge" I just referred to, and it also has logic that calls into some additional tooling in AWS that provides more of the bridging. Although I do think it's important to have some understanding about how your tools work, there's a point where it's okay to say "okay, it just works."

Let's walk through the steps that I performed to transform my API. While this article is lengthy, most of the details are here to provide a deeper understanding of the choices I've made and how things are working. But the actual steps are not that many. If you want to follow along, I've included the previous solution in the downloads for this article.

Creating the New Project

Start by creating a new project and in the template finder, filter on AWS and C#. This gives you four templates and the one to choose is AWS Serverless Application (.NET Core—C#). After naming the new project, you'll get a chance to choose a "Blueprint", i.e., a sample template for a particular type of app. From the available blueprint options, choose ASP.NET Core Web API. This is the template that includes the plumbing to ensure that your controller methods can be run behind a Lambda function. The project that's generated (shown in **Figure 1**) looks similar to the one created by the ASP.NET Core Web API template with a few exceptions.

- One exception is the introduction of the S3ProxyController. This is just a sample controller that I'll remove from my project. I'm keeping the values controller so that I can validate my API if needed.
- Another is the aws-Lambda-tools-defaults.json file. This file holds settings used for publishing whether

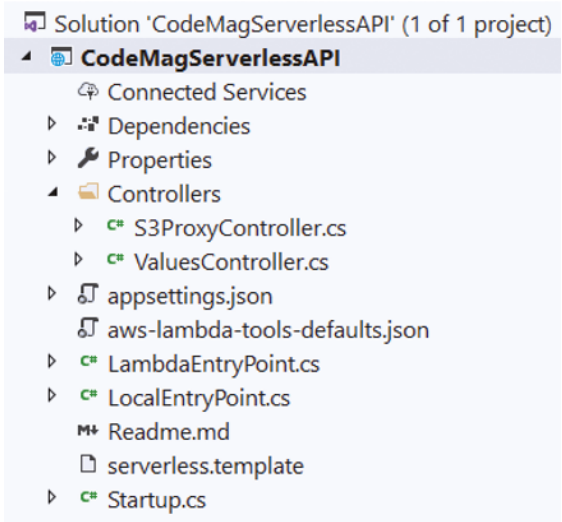


Figure 1: The project generated from the selected template

```
<ItemGroup>
  <PackageReference Include="Amazon.Extensions.Configuration.SystemsManager" Version="1.2.0" />
  <PackageReference Include="AWSSDK.S3" Version="3.3.110.44" />
  <PackageReference Include="AWSSDK.Extensions.NETCore.Setup" Version="3.3.101" />
  <PackageReference Include="Amazon.Lambda.AspNetCoreServer" Version="5.0.0" />
  <PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="3.1.3" />
  <PackageReference Include="Microsoft.EntityFrameworkCore.Tools" Version="3.1.3">
    <PrivateAssets>all</PrivateAssets>
    <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>
  </PackageReference>
 </ItemGroup>
<ItemGroup>
```

Figure 2: The EF Core and SystemsManager package references added to the project file

you are doing so via the tooling or at the command line using AWS' dotnet CLI extensions.

- LambdaEntryPoint.cs replaces program.cs for the deployed application.
- LocalEntryPoint.cs replaces program.cs for running or debugging locally.
- The serverless.template contains configuration information for the deploying the application. Specifically, this uses the AWS SAM specification, which is an AWS CloudFormation extension used for serverless applications.

Copying Assets from the Original API

Before looking at the Lambda-specific files, let's pull in the logic from the original API. In the downloads that accompany this article, you'll find a BEFORE folder that contains the solution from the previous article.

First, you'll need to add the NuGet references for the EF Core packages (SqlServer and Tools for migrations) as well as the SystemsManager extension you used for the deployed API to read the secured parameters stored in AWS. You can see the packages in the csproj file shown in **Figure 2**.

Next, I'll copy files from the previous application into the project and remove the S3ProxyController file. The files I copied in, highlighted in **Figure 3**, are the **AuthorsController**, the **BookContext**, the **Author** and **Book** classes, and the contents of the Migrations folder. As a reminder, the AuthorsController was originally created using the controller template that generates actions using Entity Framework.

Note that in the previous article, I created a SQL Server database instance in Amazon's RDS, let EF Core migrations create the database and tables, and then manually added some data via SQL Server Object Explorer. The "Before" solution that comes with this article has two changes related to the data. Its BookContext class now includes HasData methods to seed some data into the Authors and Books tables. Also, there is a second migration file, seeddata, that has the logic to insert the seed data defined in the BookContext. If you don't have the database yet, you'll be able to use the update-database migrations command to create the database and its seed data in your database instance. But you will have to create the database instance in advance.

The new Startup class has some extra logic to interact with an AWS S3 Proxy, which is then used by the S3ProxyController that you just deleted. Because you don't need that, it's safe to completely replace this startup.cs file with the

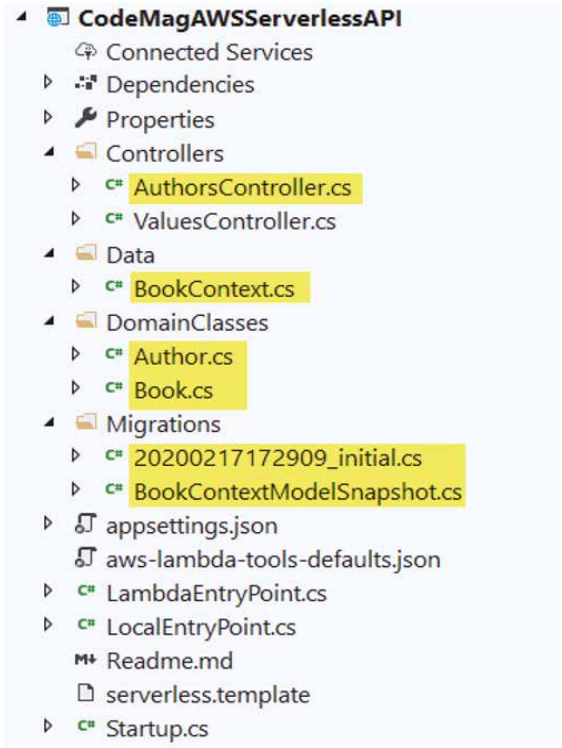


Figure 3: Files copied into the project from my original API

one from the original solution instead of copying various pieces of that file. Most important in there is the logic to build a connection string by combining details you'll add in shortly. The final bit of that logic attaches the concatenated connection string to the DbContext dependency injection configuration with this code:

```
services.AddDbContext<BookContext>
 (options =>
 options.UseSqlServer(connection));
```

My startup class (and the others) from the earlier project has the namespace, CodeMagEFCoreAPI. My new project's namespace is different. Therefore, I needed to add

```
using CodeMagEFCoreAPI;
```

Add this using statement to both the LambdaEntryPoint and LocalEntryPoint classes. The compiler will remind you about this.

Let Visual Studio Access Your Database

Note that if you're using the database you created from the previous article, you may need to re-grant access from the IP address of your PC. I hit this snag myself. The easiest way I found to do this was to view the RDS instances in the AWS Explorer, right click-on the desired instance and then choose "Add to Server Explorer." If your IP address doesn't have access, a window will open with your IP address listed and a prompt to grant access.

The next assets you need from the earlier solution are the connection string and its credentials.

The connection string to the database goes into appsettings.json, to be read by the startup class code that builds the connection string. My setting looks like this, although I've hidden my server name:

```
"ConnectionStrings": {
  "BooksConnection":
   "Server=codemagmicro.***.us-east-2.
    rds.amazonaws.com,1433;
    Database=BookDatabase"
}
```

Keep in mind that JSON doesn't like wrapped lines. They are only wrapped here for the sake of this article's formatting. Also don't forget that very important comma to separate the Logging section from the ConnectionStrings section. I can attest to how easy it is to make that mistake.

The ASP.NET Core Secret Manager will supply the DbPassword and DbUser values for the connection string at design time but won't get stored into the project, which means that you don't have to worry about accidentally deploying them. As a reminder, right-click on the project in Solution Explorer, choose Manage User Secrets, which will open a json file for the secrets. Add your secret credentials into this file, for example:

```
{
  "DbPassword": "myfancypassword",
  "DbUser": "myusername"
}
```

These values will be available to the Configuration API. With all of this in place, I'm now able to run the new API locally on my computer—hosted by .NET's Kestrel server—by choosing the project name in the Debug Toolbar. The apps read the password and user ID from the ASP.NET secrets and with those, are able to interact with my AWS hosted database.

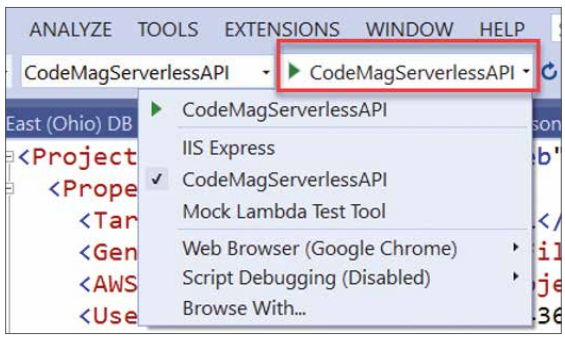


Figure 4: Targeting the project to run locally using .NET's Kestrel server

Running Locally Isn't Using Any Lambda Logic

At this point, Visual Studio isn't doing anything more than running the API in the same way as it would for an ASP.NET Core API, ignoring all of the Lambda-specific logic added by the template. The new API runs locally and the puzzle pieces are in place for this application to run as a Lambda function, but they aren't being used yet.

So far, you're seeing that your existing skills for building ASP.NET Core apps remain 100% relevant, even for testing and debugging your apps. You don't have to worry about issues related to the Lambda function getting in the way of building and debugging the app. All of that logic stays out of your way for this part of the application building. That's because the project knows to run locally from the LocalEntryPoint class, which avoids all of the Lambda infrastructure. Other than the class name, LocalEntryPoint.cs is exactly the same as program.cs in a typical ASP.NET Core API project. And by default, the debugger will start by calling its Main method.

Understanding How Your API Will Transform into a Lambda Function

With my API now running successfully, it's time to trigger the special logic included in the template to run all of this as a Lambda function.

I think it's important to understand some of the "magic" that is happening for this scenario. Of course, it's not magic. It's some very clever architecture on the part of the AWS Lambda team. Keep in mind that the biggest difference between running the API as a regular Web application and running it as a serverless application is that the Web application is always running and consuming resources, whereas the serverless application is a Lambda function that acts as

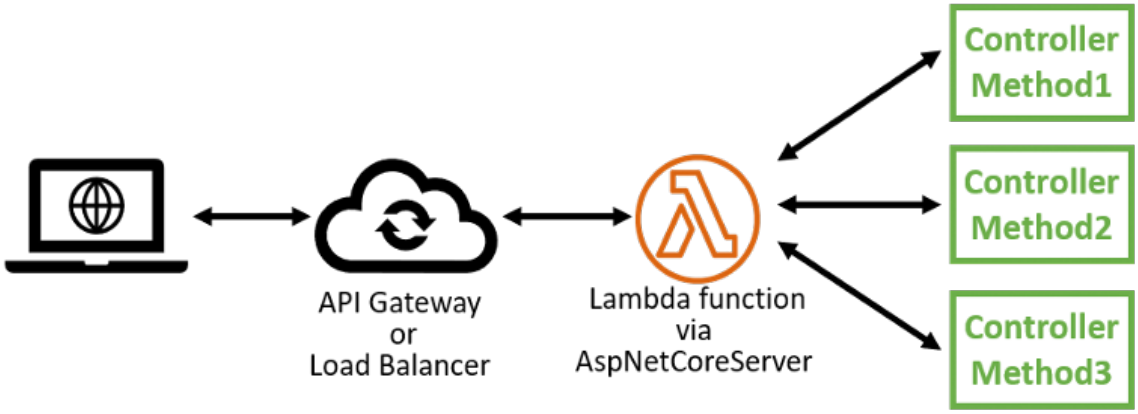


Figure 5: How your hosted API works after being transformed during deployment

Watch Out for Publicly Available Production Databases

Keep in mind that when originally creating the database instance (in the earlier article), I specified that it should be publicly available which, combined with setting accessibility to my development computer's IP address, allows me to debug the API in Visual Studio while connecting to the database on AWS. I can also connect through Visual Studio's database tools, SSMS, Azure Data Studio or other tools. However, for a production database that's being accessed by another AWS service (e.g., this Lambda function app, or my API deployed to AWS Elastic Beanstalk) you should disable the public availability for the instance. For testing, you could have one test database in its own publicly available instance (still limited to select IP addresses) and leave the production database locked down.

a wrapper to your controller methods. And by running on demand, that means you are only paying for the resources used in the moments that function is running, not while it's sitting around waiting for a request to come in.

When the app is deployed (using some of the special assets added by the template) it doesn't just push your application to the cloud, it builds a full Lambda function infrastructure. Because this function is meant to be accessed through HTTP, it's shielded by an API Gateway—the default—but you have the option to switch to an Application Load Balancer instead. Unlike a regular ASP.NET Core API, the controller methods aren't exposed directly through URIs (or routing). A Lambda function wraps your controllers and runs only on demand when something calls your API. If nothing calls, nothing is running.

What's in between the gateway and your controller is the Amazon.Lambda.AspNetCoreServer, which contains its own Lambda function that translates the API Gateway request into an ASP.NET Core request. The requests to that Lambda function are all that you pay for in this setup, not the controller activity; that is, after the monthly free allocation. There's more to how this works but for the purposes of this article, this should be enough to have a high-level understanding of what appears to be magic.

Engaging the SAM and Lambda Logic in Your API

So now let's take a look at some of the assets shown in **Figure 1** that were created by the template. Your friend here is the Readme markdown file included in the project. It gives some insight into the assets and I will highlight some of the relevant descriptions here for you:

- **serverless.template**: an AWS CloudFormation Serverless Application Model template file for declaring your Serverless functions and other AWS resources
- **aws-Lambda-tools-defaults.json**: default argument settings for use with Visual Studio and command line deployment tools for AWS
- **LambdaEntryPoint.cs**: class that derives from Amazon.Lambda.AspNetCoreServer.APIGatewayProxyFunction. The code in this file bootstraps the ASP.NET Core hosting framework. The Lambda function is defined in the base class. When you ran the app locally, it started with the LocalEntryPoint class. When you run this within the Lambda service in the cloud, this Lambda EntryPoint is what will be used.

In addition to the Using statement mentioned above, there is one more change to make in the LambdaEntryPoint class. If you read the earlier article, you may recall that there was also a lesson in there on storing the database UserId and password as secured parameters in AWS. I was able to leverage the SystemsManager extension to read from AWS Systems Manager where the parameters are stored. You'll need to ensure that the deployed app can do that by adding the following builder.ConfigureAppConfiguration code into the Init method of the LambdaEntryPoint class. This will also require a Using statement for Microsoft.Extensions.Configuration.

```
protected override void Init
 (IWebHostBuilder builder)
{
  builder.ConfigureAppConfiguration(
   (c, b) =>
    b.AddSystemsManager("/codemagapi")
  );
  builder.UseStartup<Startup>();
}
```

In the serverless.template file, you also need to make a simple change to the policies controlling what the function can access, so that it can read the parameters.

By default, AWS' AWSLambdaFullAccess policy is defined directly in the serverless.template without using roles. You can see this in the Properties section of the AspNetCoreFunction resource in the file:

```
"Role": null,
"Policies": [
    "AWSLambdaFullAccess
],
```

You just need to add two more policies, AmazonSSMReadOnlyAccess and AWSLambdaVPCAccessExecutionRole. The Role property is not needed at all, so I removed it.

```
"Policies": [
    "AWSLambdaFullAccess",
    "AmazonSSMReadOnlyAccess",
    "AWSLambdaVPCAccessExecutionRole"
],
```

The SSM policy gives the deployed function permission to access the parameters in the Systems Manager. The VPCAccess policy gives the function permission to wire up a connection to the VPC that's hosting the database. I'll point out when this comes into play after deploying the function.

There is some more cleanup you can do in the serverless template. Many settings in there are related to the S3Proxy controller that you deleted. You can delete the related sections.

The sections you can delete, starting from the top are:

- Parameters
- Conditions
- The Environment section within Resources: AspNetCoreFunction
- Bucket within Resources
- S3ProxyBucket within Outputs

Take care to get correct start and end points when deleting sections from this JSON file, including commas.

A related setting in appsettings.json is the AppS3Bucket property. You can delete that as well.

Deploying the Serverless Application

Although you can run the non-Lambda version of the app locally as I did earlier, you can't just install the Lambda service on your computer to check out how it works with the infrastructure. You need to deploy the application to AWS. This is a simple task, thanks again to the toolkit. Let's walk through that process.

The Relationship Between the Function and the Database's VPC

At first, I thought that specifying a VPC for the application meant that I would be pushing the app into the same VPC as the database. This misunderstanding led me on a wild goose chase. I started by creating a separate VPC and could never get it to communicate with the database. It's important to understand that you aren't creating a VPC for hosting the application, but identifying the existing VPC with the subnet(s) that allow the access to the RDS instance. It's like you're making an introduction between the VPC and the Systems Manager.

The aws-Lambda-tools-default.json file contains configuration information for publishing the function. In fact, the file also has configuration information for creating the S3 Proxy used by the controller which we have now deleted. Deleting the "template-parameters" property from the file will clear that extraneous information.

The context menu for the serverless project has the option: "Publish to AWS Lambda...". This triggers a form to open where you can specify settings for your deployed application. The profile and region are pre-populated using your AWS Explorer settings.

You'll need to name the CloudFormation stack and bucket for the deployment. All of the resources for your application will be bundled up into a single unit and managed by CloudFormation. This is what is the stack refers to. The S3 bucket (different from the S3 Proxy used by the deleted controller) will store the application's compiled code for your function. Any existing buckets in your account are listed in the dropdown, and you can create a new one with the New button.

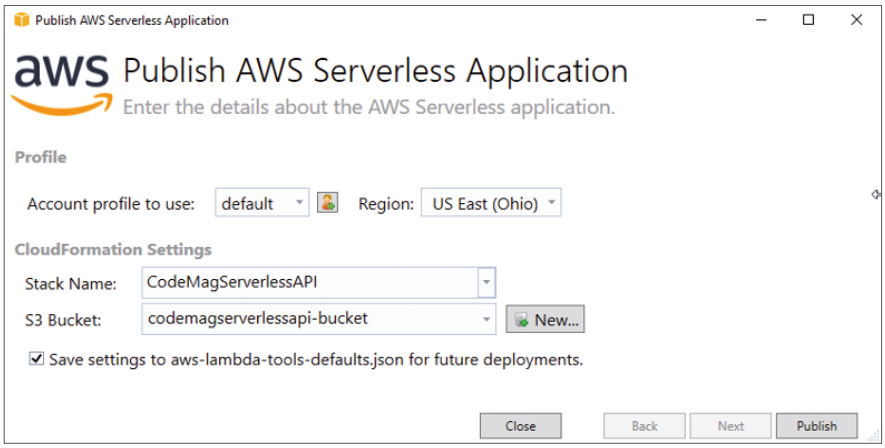


Figure 6: Publishing the serverless application

Note that bucket names must be all lower case. Also note that if you selected an existing bucket, it needs to be one that's in the same region as the one where you're deploying the Lambda function. My settings are shown in **Figure 6**.

Now you're ready to publish the application, so just click Publish. You'll immediately start to see log information about the steps being taken to build and push the application to the cloud. After that, a log is displayed showing what's happening in the cloud to create all of the infrastructure to run the application. When it's all done, the status shows CREATE_COMPLETE and the final logs indicate the same. (**Figure 7**).

The URL of the application is shown on the form. Mine is https://hfsw7u3sk5.execute-api.us-east-2.amazonaws.com/Prod.

Sending a request to the values controller will succeed, in my case at https://hfsw7u3sk5.execute-api.us-east-2.amazonaws.com/Prod/api/values, but the authors controller will fail with a timeout. That's because you have a bit more security configuration to perform on the newly deployed function.

Let's first look in the portal to see what was created and then address the last bits of security for the authors controller in the AWS cloud to access the authors data in the database.

Examining What Got Created in the Cloud

If you refresh the AWS Lambda node in the AWS Explorer, you should see your new function app listed. Its name will start with the CloudFormation stack you specified in the publish wizard, concatenated with "AspNetCoreFunction" and a randomly generated string. You can update some of the function's configuration, look at logs, and more. You might notice the Mock Lambda Test Tool in the toolkit. But this is not for debugging the cloud-based Lambda from Vi-

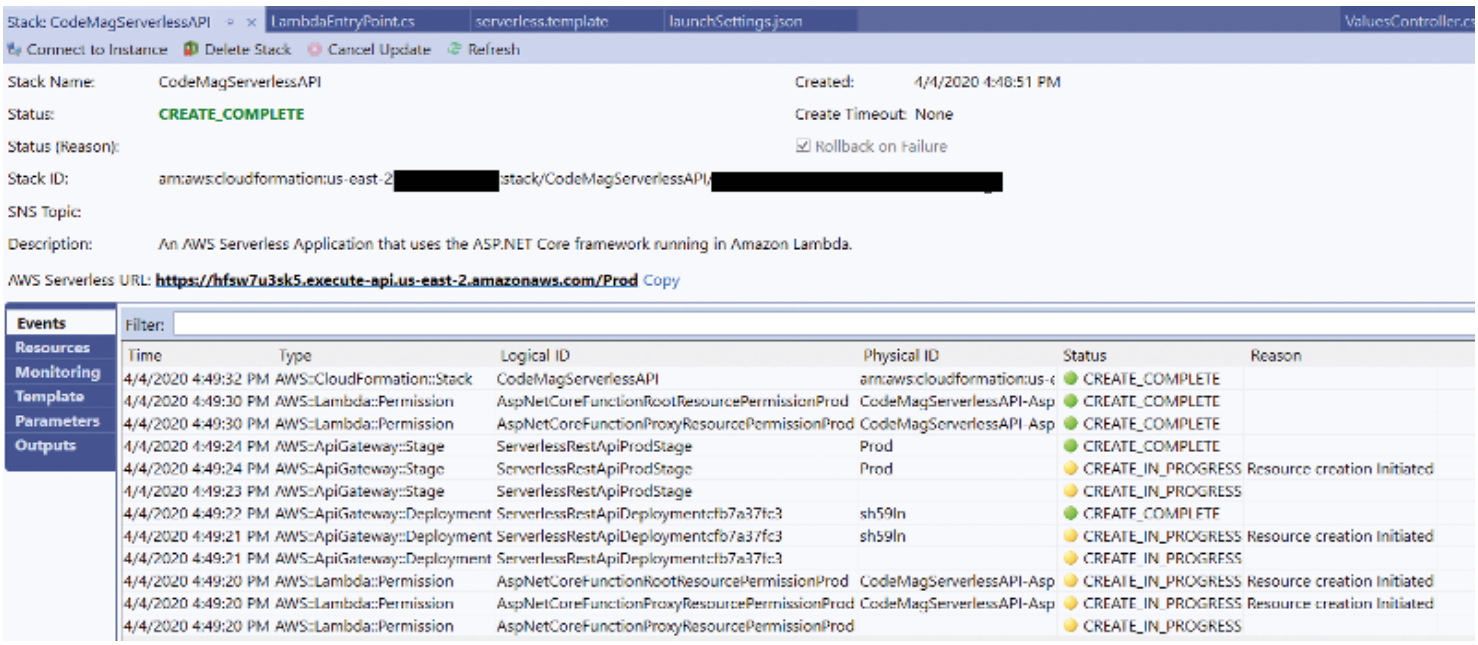


Figure 7: The final logs when the deployment has completed

sual Studio. It's for performing an advanced form of testing to debug problems in the deployed function. You can learn more about that tool here: https://aws.amazon.com/blogs/developer/debugging-net-core-aws-Lambda-functions-using-the-aws-net-mock-Lambda-test-tool/. I'll come back to the configuration page shortly.

When learning I like to also see the function in the portal. It feels more real and more interesting to me. Here's how to do that.

Log into the portal and be sure to set your view to the region where you published the function. Select Lambda by dropping down the Services menu at the top. From the AWS Lambda dashboard, select the Functions view. Here, you'll see the same list of Lambda functions in your account, filtered by whatever region is selected at the top of the browser page.

Click on the function to open its configuration page. At the top, there's a note that the function belongs to an application with a link to see some information about the application: the API endpoint and a view of the various resources (your Lambda Function, an IAM role associated with the function, and the API gateway). There are other details to explore in the application view, such as a log of deployments and monitoring.

Back in the function's overview page, the first section shows a visual representation of the function with an API gateway block and the function itself. Click on the API gateway to see the two REST endpoints that were created: one with a proxy and one without. Next, click on the block for the function and you'll notice that the display below changes. If your app was using a scripting language, there would be a code editor available. Because you're using a language that requires a compiler, uploading a zip file is the only option—and that's what the Publish wizard did for you —so the code editor is hidden. Keep scrolling down to see more sections: Environment variables, Tags, and a few others.

The block of interest is the currently empty VPC area. VPC is an acronym for Virtual Private Cloud, a logically isolated section of the AWS cloud. The VPC settings are the key to giving the function permission to access the database instance. Currently, the lack of that access why the authors controller is failing.

Understanding and Affecting What Permissions the Function Has

Thanks to the AmazonSSMReadOnlyAccess policy you added to the function in the serverless.template file, the function is able to access the UserId and Password parameters you stored in the Systems Manager as part of the previous article. However, even though it can read the connection string credentials for the database, it isn't able to connect to the VPC where the database lives. Everything is secure by default here. Even from other services attached to the same IAM account.

The database instance is inside the default VPC in my AWS account. That's most likely the case for you if you followed the demo in the earlier article. The function itself isn't inside a VPC. As I explained earlier, it was deployed to a CloudFormation stack, the one you named in the Publish wizard. What you need to do next is tie the function to the VPC that

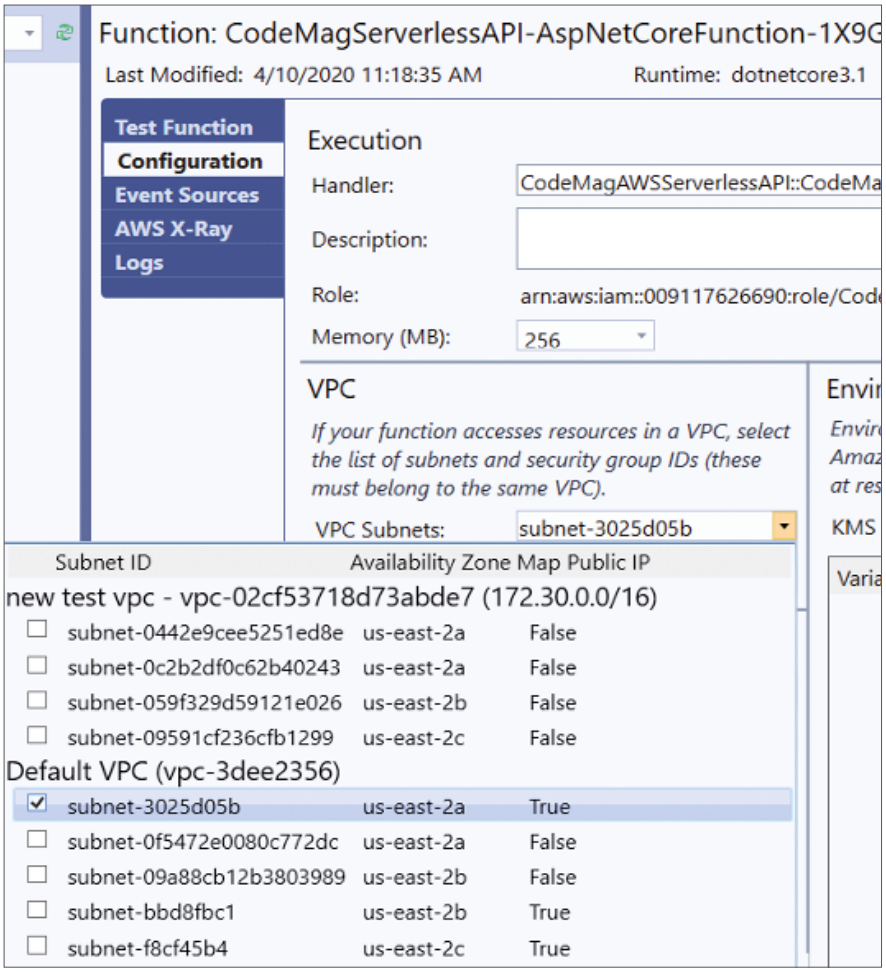


Figure 8: Selecting a subnet within the database's VPC

contains the database instance. You can do that through the portal or using the Function configuration page of the Toolkit in Visual Studio. I'll show you how to do this back in Visual Studio.

The Toolkit's function configuration page has a VPC section and in there, a drop-down to select one or more VPC Subnets to which you can tie the function and a drop-down for security groups. The latter is disabled because it only shows security groups for the VPC(s) you've selected.

A subnet is essentially a part of an IP range exposed through the AWS cloud. A VPC can have one or more subnets associated with it. By default, the default VPC has three subnets and each of those is a public subnet, meaning that it's allowed to receive inbound requests from the Internet (and can make outbound calls), e.g., a Web server. To connect the Lambda function to this VPC, you can select any one of the subnets from the default VPC. If the VPC has private subnets, connecting to one of those will work as well. Based on all of my experiments (and guidance specifically for this article from experts at AWS), you can randomly choose any subnet attached to the VPC as I've done in **Figure 8**. Note that the "Map Public IP" column isn't an indication of whether the subnet is public or private.

Having selected that subnet, the Security Groups drop-down now gives me two security group options—these are the

only groups tied to that VPC. If you have more, they'll all be available in the drop-down. There will always be a default security group, so you can select that one.

Once you've specified the subnet and security group, save the settings by clicking the "Apply Changes" icon at the top of the Function page. Unfortunately, the toolkit doesn't provide status. So, I flipped back to the portal view, refreshed the Web page and waited for the message "Updating the function" to change to "Updated". The message is displayed right at the top of the page in a blue banner so shouldn't be hard to find. This took about one minute.

Remember adding the AWSLambdaVPCAccessExecutionRole to the serverless.template policies earlier? That policy is what gave the Lambda function permission to perform this action of attaching to the VPC.

One Last Hook: VPC, Meet Systems Manager

Now, if you test the api/values again or the api/authors, you may think you've broken everything! Both controllers time out. But you haven't broken the function. The function itself is able to access the parameters, but the VPC is not. Therefore, now that the function has been configured to run attached to my VPC, it can't reach back to Parameter Store over the Internet. Recall the logic you added to the LambdaEntryPoint class:

```
builder.ConfigureAppConfiguration((c, b)
 => b.AddSystemsManager("/codemagapi"));
```

That's the very first line of code executed by the application, and since the VPC isn't allowed to reach the Systems Manager by default, that code fails before even trying to run any controller code.

The final piece of the puzzle is to allow the VPC access to the Systems Manager. There are two options. One is to configure the VPC to allow the Lambda function to go out to the Internet and then to the service for the Parameter Store. The other is to configure a channel (called an endpoint) on the VPC that allows the function to call the Systems Manager without ever leaving the AWS network. The latter is the simplest path and the one I chose.

I'll create endpoint on the default VPC, giving the endpoint permissions to call the Systems Manager. Endpoints aren't available in the toolkit, so you'll do that in the portal, and luckily, it's just a few steps where you can rely mostly on default settings. It's not a bad idea to get a little more experience with interacting with the portal. Alternatively, you could do this using the AWS CLI or AWS' PowerShell tools as well.

In the portal, start by selecting VPC from the AWS Services list. From the VPC menu on the left, select Endpoints, then Create Endpoint. Filter the available service names by typing ssm into the search box, then select com.amazonaws.[region].ssm.

From the VPC drop-down, select the relevant VPC. It's handy to know the ID of your VPC, or its name, if you've assigned one in the console. Once selected, all of that VPC's public subnets are preselected, which is fine. In fact, all of the rest of the defaults on this page are correct, so you can scroll to the bottom of the page and click the Create endpoint button.

That's it! The endpoint should be ready right away. The application now has access to the parameters, and it's able to use those parameters to build the connection string and access the database. Finally, the api/values and api/authors should successfully return their expected output.

A Journey, But in the End, Not a Lot of Work

Although this article has been a long one, so much of what you read was to be sure you understood how things work and why you were performing certain steps. In fact, the journey to modernize your ASP.NET Core API to AWS Lambda functions doesn't entail a lot of work and the value can be significant.

You created a project from a template, copied over files from the original API and made a few small changes to a handful of files. With this, the API was already able to run locally in Visual Studio.

Then you used a wizard to publish the API to AWS as a Lambda function and because the API interacts with a SQL Server database in Amazon RDS (using Entity Framework Core), you needed to enable a few more permissions. That was only two steps: Connect the database's VPC to the function and create an endpoint so that VPC was able to access the credentials that are stored as AWS parameters.

Although this exercise was focused on an existing API, you can also just create a function app from scratch with this template and, using the new project, build an ASP.NET Core API from scratch using all of the knowledge you already have for doing that without having to learn how to build Lambda functions. Surely, like me, once you've whetted your appetite with this, you'll probably be curious and ready to explore building straight up Lambda functions next.

Julie Lerman

CODE

Break into the Tech Industry: Enroll in an Online Tech Academy Coding Boot Camp today!

The Tech Academy is a technology school that delivers online training, with students all over the world. In fact, they just received 2020's Best Online Coding Bootcamp Award.

Learn Coding Anywhere

The Tech Academy programs require no technical background or coding experience. Our classes are deisgned for absolute beginners. We specialize in coding boot camps that train students in a wide range of technology subjects, including:

- Website development
- Computer programming
- Design
- Data science
- And more...

These programs can be sompleted in as little 8 weeks and prepare graduates for working in the tech industry.

The Tech Industry is Still Hiring

Despite recent global events, we are placing a high number of graduates in remote, technical positions. While many companies have been struggling, the tech industry has been impacted less than others.

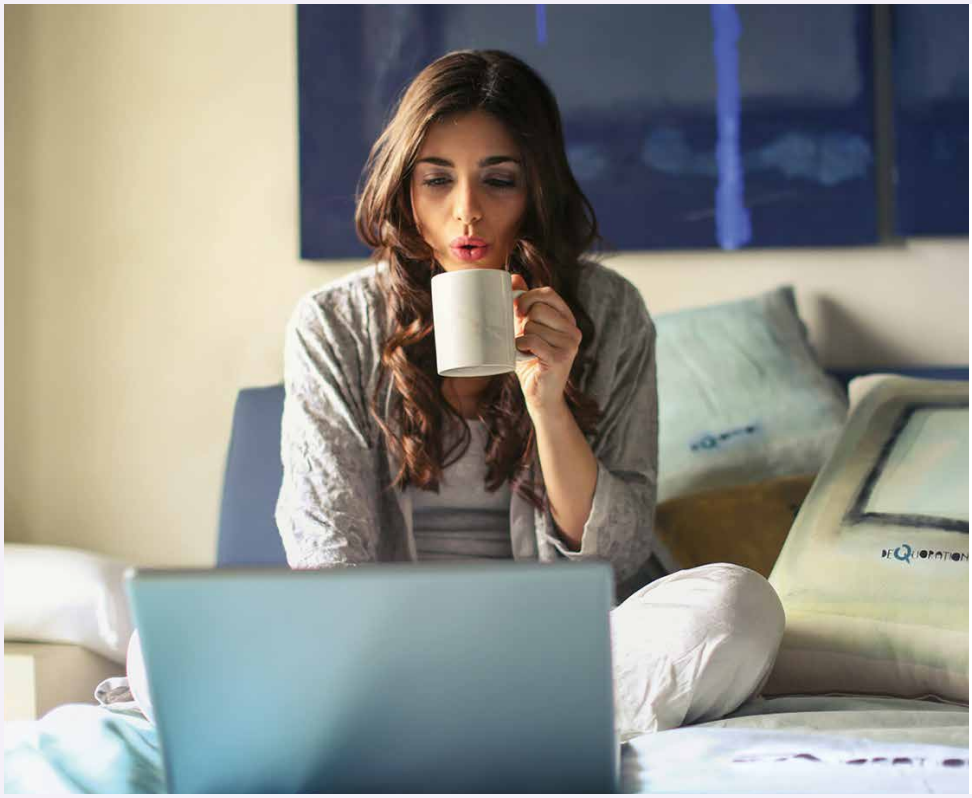
During these uncertain times, one thing is for sure: the need for technology isn't going anywhere – in fact, it's increasing.

Prepare for your future today by completing a Tech Academy Boot Camp from the comfort an safety of your own home.

What Sets the Tech Academy Apart

There are other online schools, so what makes The Tech Academy special?

- No tech background or coding experience is required
- Online and in-person training
- Open enrollment—start anytime
- Flexible scheduling options
- Self-paced program
- Outstanding job placement assistance
- Multiple financing options



Start Your Career Path Today

The Tech Academy is enrolling now!

Contact us at:
info@learncodinganywhere.com

Or call: (503) 206-6915

Power Query: Excel's Hidden Weapon

You've certainly worked with Excel extensively. You know its ubiquity across the modern business world. Excel examples range from debt forecasting and pricing models to enable decision-making to ad hoc accounting reports. I absolutely embrace its versatility, its incredible calculations capabilities, and its stable presence in an everchanging business world. However, it still

Helen Wall

www.linkedin.com/in/helenrmwall/
www.helendatadesign.com

Helen Wall is a power user of Microsoft Power BI, Excel, and Tableau. The primary driver behind working in these tools is finding the point where data analytics meets design principles, thus making data visualization platforms both an art and a science. She considers herself both a lifelong teacher and learner. She is a LinkedIn Learning instructor for Power BI courses that focus on all aspects of using the application, including data methods, dashboard design, and programming in DAX and M formula language. Her work background includes an array of industries, and in numerous functional groups, including actuarial, financial reporting, forecasting, IT, and management consulting. She has a double bachelor's degree from the University of Washington where she studied math and economics, and also was a Division I varsity rower. On a note about brushing with history, the real-life characters from the book The Boys in the Boat were also Husky rowers that came before her. She also has a master's degree in financial management from Durham University (in the United Kingdom).

can instill fear and anxiety of running into potential problems like finding undetectable mistakes in large files or unknowingly using an outdated data set. For those reasons, Excel often becomes the target of ire in the analysis process rather than the users who perhaps incubated the problem to begin with.

You might ask what better options exist out there to manage these large data sets? The answer comes by utilizing Power Query. I'm going to show you how it alleviates many of these concerns by making the entire ETL framework much more efficient and scalable. It's an Excel game-changer.

What Is Power Query?

What exactly is Power Query? It's a tool that sits inside Excel (as well as Power BI) that enables you to automate the ETL process of bringing data into Excel. This query editor extracts data from different data sources, transforms this data, then finally loads it into Excel where you can create additional calculations and modeling. Perhaps more importantly, if you want to repeat this process of bringing in the same data every month, you can refresh the entire ETL framework with the click of a button. Power Query provides the ideal bridge between organization segments because the business will find that it's often an easy process to set up and maintain, and developers in turn can reduce their scheduling obligations to maintain these types of reports.

Where can you get your data from? Power Query probably has at least one connection that works for you (plus many more) including:

- SQL Server
- SAP
- Azure applications
- Amazon Web Services
- Web API queries

Once you connect to the data source, you can then transform the data connection into a useful data set by employing a plethora of built-in query editor interface functions like:

- Trimming spaces
- Splitting a column into several columns
- Creating new calculations
- Grouping fields
- Pivoting to reshape data

After applying these transformation steps, you then load this clean, well-structured data table directly into Excel. Concerned about a data update later today that could change all your numbers? Simply refresh the entire ETL framework with the click of a button.

Power Query allows business users to develop reports with current technology that uses no code, and it allows developers to create something tangible for the business side. It's easy to access, requires very little coding, and is easy to refresh. It's a scalable solution that can keep up with the demands of your organization without requiring a massive investment of time and energy to set up the process that you would face with a larger solution like setting up the data in an enterprise-wide database.

Our Challenge: Converting Currencies

Let's say you work for a company that does business in both the US and the European Union. In addition to worrying about differing rules and laws for business in multiple countries, you also need to worry about the conversion rates between these two monetary regions. Let's say, for the sake of this project, that you want to convert these currencies into US Dollars for your financial reports.

You need to obtain the currency data first, and you can get it from the Federal Reserve of Economic Data (FRED) for St. Louis, a free resource that allows you to not only view the trends for key economic and finance data, but also download it or connect to its API. Here's the link to the FRED webpage (**Figure 1**): https://fred.stlouisfed.org/series/DEXUSEU

Remember that you want to make the process of getting data more efficient. Downloading the data before connecting to it adds steps to the process rather than reducing them, which in turn increases completion time and the potential for making mistakes. There's not typically one right answer to solving a problem but using the FRED API is both easy to set up and refresh in future months.

Before you jump into working in Excel, you need to first get access on the FRED website:

1. Navigate to the FRED account sign-up page to get your own account if you don't already have one. Remember your username and password; you will need them later to configure your sign-in credentials in Power Query.
2. Because you'll be using the API query connection to the FRED data, you'll need to sign up for your own unique API key through the API menu and requesting a new key (**Figure 2**). Once you add this API key, you'll see it appear on this page.

Accessing Power Query

For those familiar with Microsoft's Power BI Desktop, you know that initially opening the application immediately prompts you to select your data source. After selecting a data connection type, it takes you into the Power Query Editor to begin the transformation process and add any additional queries. Excel offers similar ETL capabilities in its own Power Query Editor. Unfortunately, when you open Excel, it doesn't prompt you to set up a data connection, and thus few business users even know this incredible tool exists.

To access Power Query in Excel:

1. From the Data menu, choose the Get Data menu item.
2. You will next see the prompt to choose your data source

Figure 1: FRED website data

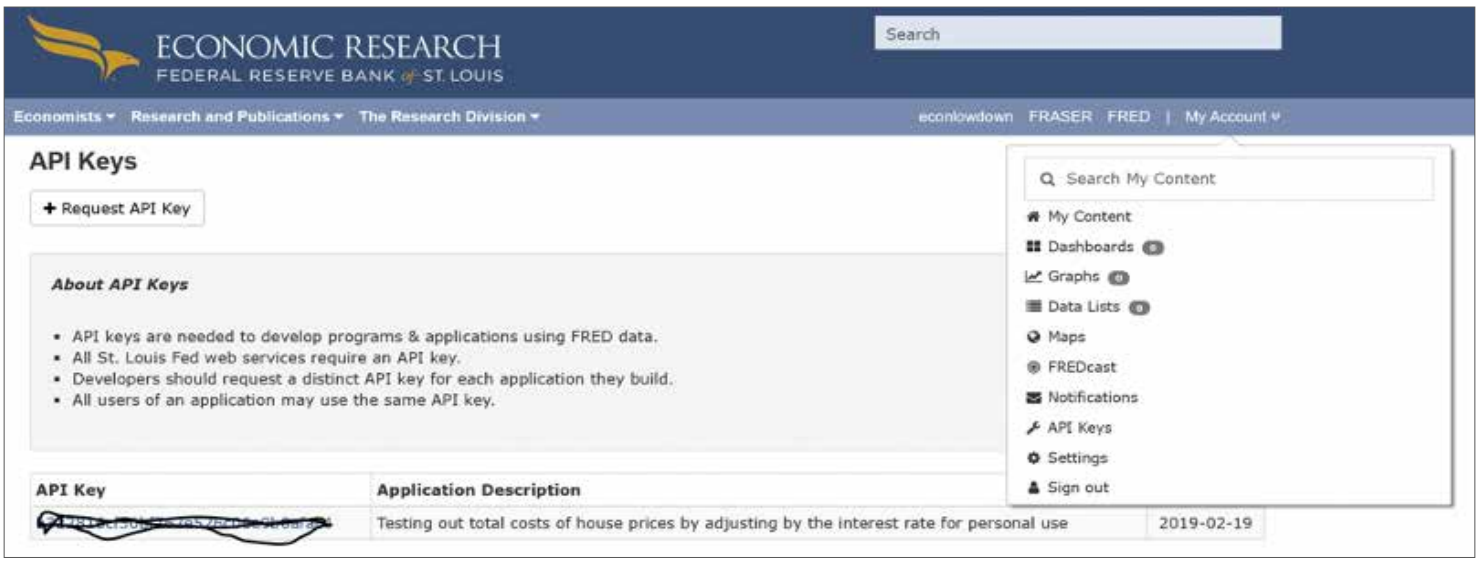


Figure 2: API key

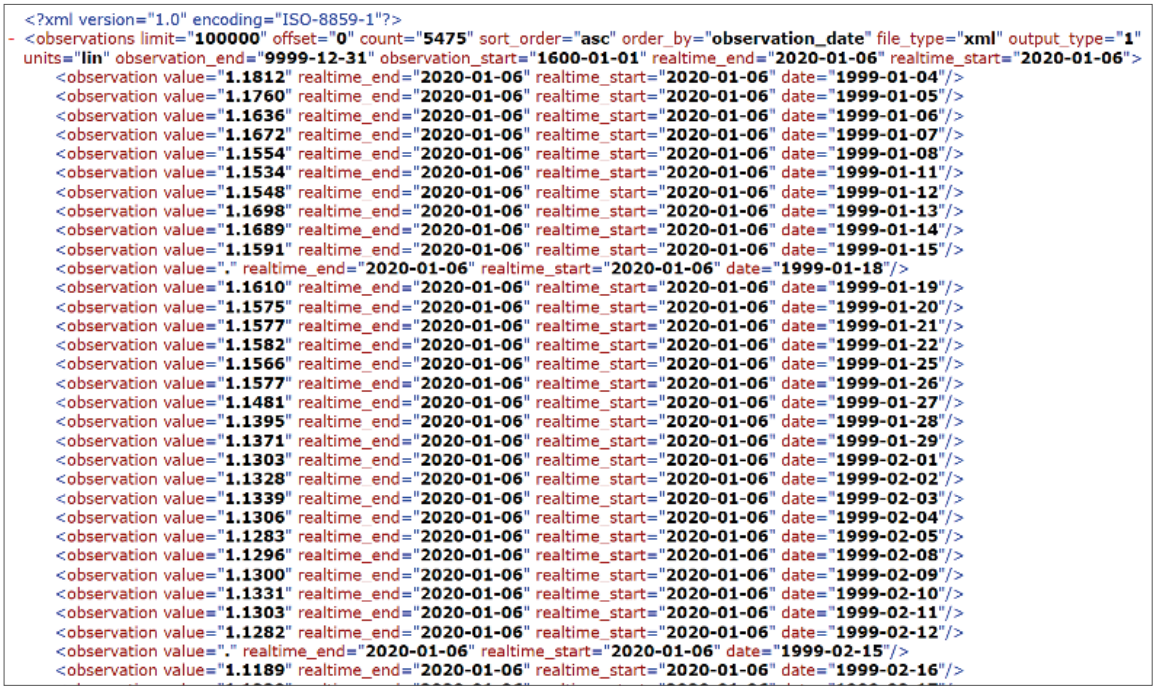


Figure 3: API query setup

from a large array of data connectors, such as SQL Server, another Excel file, or a Web connection. You can also select Launch Power Query Editor.

Note that you can only access Power Query in the later Excel versions, so for those of you working with earlier Excel versions, you must update to a later version! Once you create a new query and load it to Excel, you can open Power Query again by going back into the query editor. You can also refresh the data directly in the Excel interface without opening the query editor.

Making Power Query Work

You want to create an elegant ETL process in Power Query to first bring in the data, then later refresh it with a single button. You'll just see just a few key concepts within the Power Query capabilities in this example. When you create your own queries, you can leverage many more existing functionalities.

Step 1: Extract the Data

For this project, you're going to obtain the FRED US-Euro exchange rate data by querying the FRED API. This API connection works with a lot of data series, and it's also quite straightforward to configure. You set up the FRED API query as a single URL string comprised of three components: the API endpoint, the query parameters, and your own unique API key. I typically test out API connections and queries in a resource like Swagger Inspector (**Figure 3**). Although Power Query is an incredibly useful tool for working with data, it's unfortunately not a compiler and doesn't give much feedback on errors, unlike tools like the Swagger Inspector. Remember to replace the sample API key from the documentation with your own unique API key. From the FRED

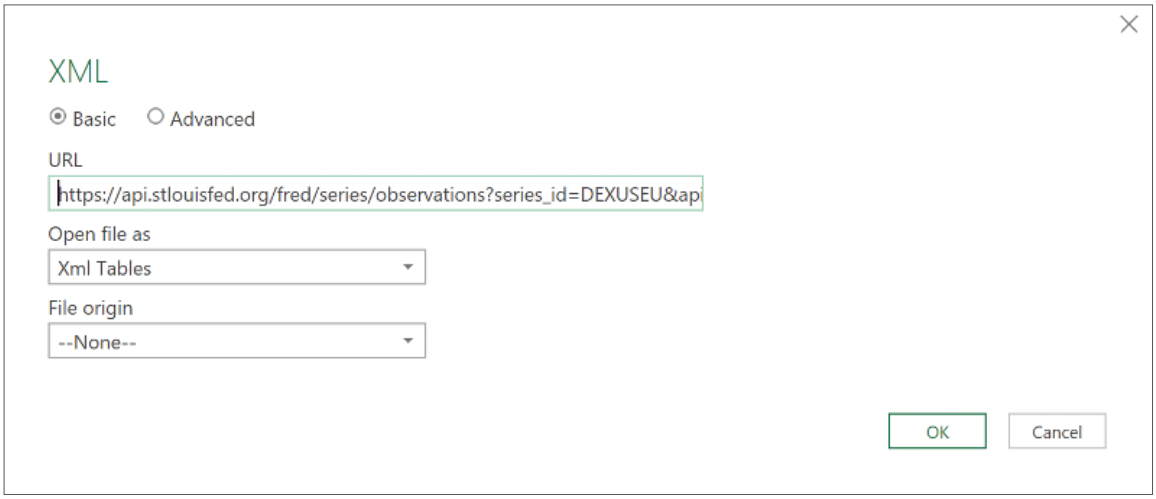


Figure 4: XML table for the API query

website, you can learn more about how to set up these queries in the API documentation webpage. Here's a link to try your own API key: https://api.stlouisfed.org/fred/series/observations?series_id=DEXUSEU&api_key=yours

You may recognize the format of the API query results as the XML data structure. Power Query can handle API query results returned in many different other formats as well, such as JSON data structures. There are limitations for connection to Web data like this in Power Query, but it's an efficient tool to create a simple API connection like this.

To configure the API query connection in Power Query:

1. Select the Web connection option.
2. In the dialog box that opens, paste the URL string for the API connection with your own API token into the URL text box (**Figure 4**).
3. Notice that you can also select advanced options for the Web connection, like splitting up the query into several pieces (splitting up the parameters for example) or using the API key or token as a header.
4. Confirm these selections, and you'll see the query editor process the connection by importing it.

As part of the extract process, you can set up log-in credentials specific to the FRED website because the API query only works with an active login. Passing these credentials into Power Query enables you to refresh the entire query without logging into the FRED website. Here's how:

1. Select the Source Settings option, which opens a new window.
2. Select the FRED API query to edit the query permissions.
3. In the Basic authorization selection input the username and password for your FRED account (**Figure 5**).

After setting up the connection, you see the query results in the middle of the screen. On the left side, you see a query list where you can add additional queries. On the right side of the screen, you see the applied steps, which will tell you the functions you perform on the query during the ETL process. You see a Source step automatically added to these applied steps list. Double-click on the gear wheel next to the step name, which opens the source connection details you see in **Figure 5**. Power Query uses a bit of AI capability to determine on its own to read this Web connection in the XML data structure. Why, then, don't you see the results returned in a structured table format? You'll see in the next step how to transform this initial connection into a useful data table.

Step 2: Transform the Data Connection

Power Query returns the API query results in the XML structure format with a single record or row of data consisting of several columns providing the connection metadata, as you see in **Figure 6**. Although the metadata provides key information about the data, like time stamps, you want to focus in on the Table hyperlink in the observation column, which represents a Power Query table object. Power Query objects consist of a combination of variables, functions, and data structures. They enable transformation capabilities such as drilling into or expanding data tables. In this example API query, the table object represents the data returned from the API query in the XML format.

If you work in Power Query a lot, you'll encounter table objects frequently and will find them immensely useful. You can access the data in table objects in a few different ways:

- Click into the Table hyperlink, which enables you to drill directly into the table. Notice that you no longer see the metadata columns in this view and you instead see the rows and columns with the table object you just drilled into.
- You will also notice a diverging arrow button next to the column name. Choosing this button opens a selection menu to select the columns to expand within the table object. Because you're not drilling into a single table object, you can still see the metadata in the view. For queries returning multiple records with table objects, expanding the table objects retains all the data in each table object, and not just the data in a single table object.

Drilling into the table object transforms the returned API query data into a readable data format, with four columns in this new view. You see a new applied step on the right side of the screen. Of these four columns, I only want to keep the attribute col-

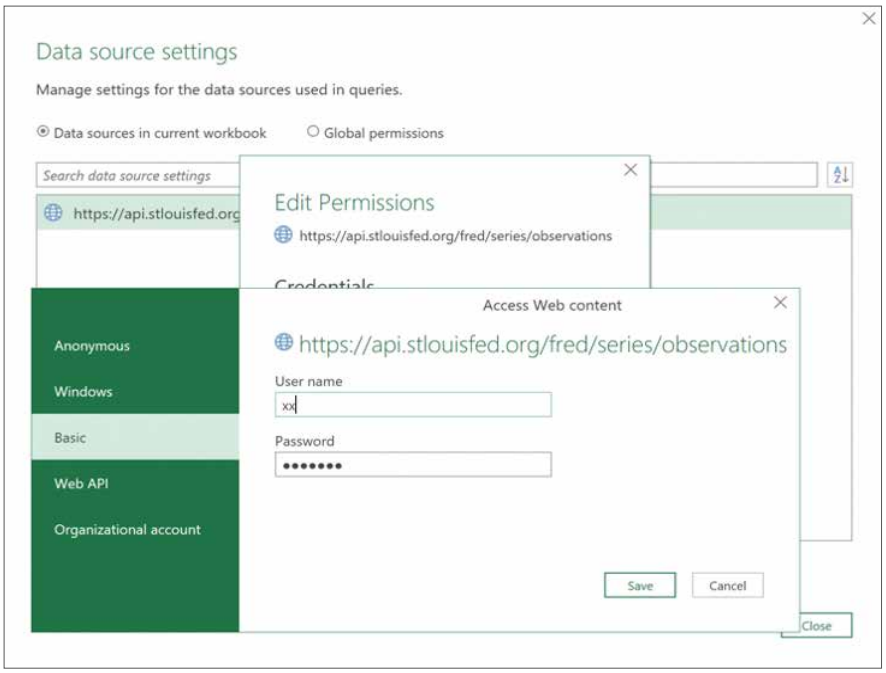


Figure 5: Source settings for FRED log-in

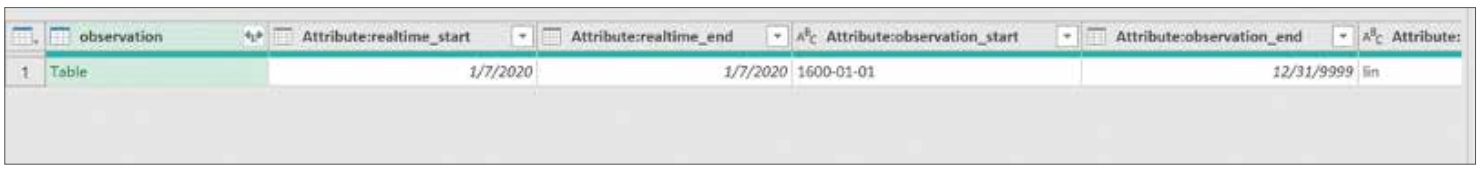


Figure 6: Initial table object

	Date	1.2 Daily Exchange Rate
1	1/4/1999	1.1812
2	1/5/1999	1.176
3	1/6/1999	1.1636
4	1/7/1999	1.1672
5	1/8/1999	1.1554
6	1/11/1999	1.1534
7	1/12/1999	1.1548
8	1/13/1999	1.1698
9	1/14/1999	1.1689
10	1/15/1999	1.1591
11	1/19/1999	1.161
12	1/20/1999	1.1575
13	1/21/1999	1.1577
14	1/22/1999	1.1582
15	1/25/1999	1.1566
16	1/26/1999	1.1577
17	1/27/1999	1.1481
18	1/28/1999	1.1395
19	1/29/1999	1.1371
20	2/1/1999	1.1303
21	2/2/1999	1.1328
22	2/3/1999	1.1339
23	2/4/1999	1.1306
24	2/5/1999	1.1283
25	2/8/1999	1.1296

Figure 7: Transformed data table

umns for the date and value because the other columns serve as metadata rather than useful data. You can select the first two columns at the same time and delete them together. You can then rename the columns to date and monthly exchange rate respectively to make it more readable. Power Query also gives you the capability to make changes to a previous step that ultimately update the rest of the query. If you select the step before the step to change the data type, you see that these rows throwing errors use a period in place of an empty field to represent closed markets on holidays. When you attempt calculations with this data later, you'll run into problems. To remove the errors from this field, select Remove Errors from the menu in the field name, which gets you to the updated data set (**Figure 7**).

In this query, each applied step takes the result from the applied step before it and applies the current transformation step on top of it. If you open the Advanced Editor, you can see how Power Query applies these transformation steps through the functional M language by automatically writing out the steps. You can create some impressive custom queries with a little background knowledge of M. Start by first slightly tweaking the existing applied steps in the Advanced Editor, and then transition to creating more complex M code.

Step 3: Aggregate the Calculations

You can change the shape and orientation of a data table by leveraging several existing functionalities in the query editor. The grouping functionality lets you determine a field in the existing table to configure as a dimension in the new table shape. You can aggregate a numeric field by summing, averaging, or counting it over the grouping dimension. Let's say that you want to calculate the average exchange rate for the entire month rather than the daily exchange rate. Before applying

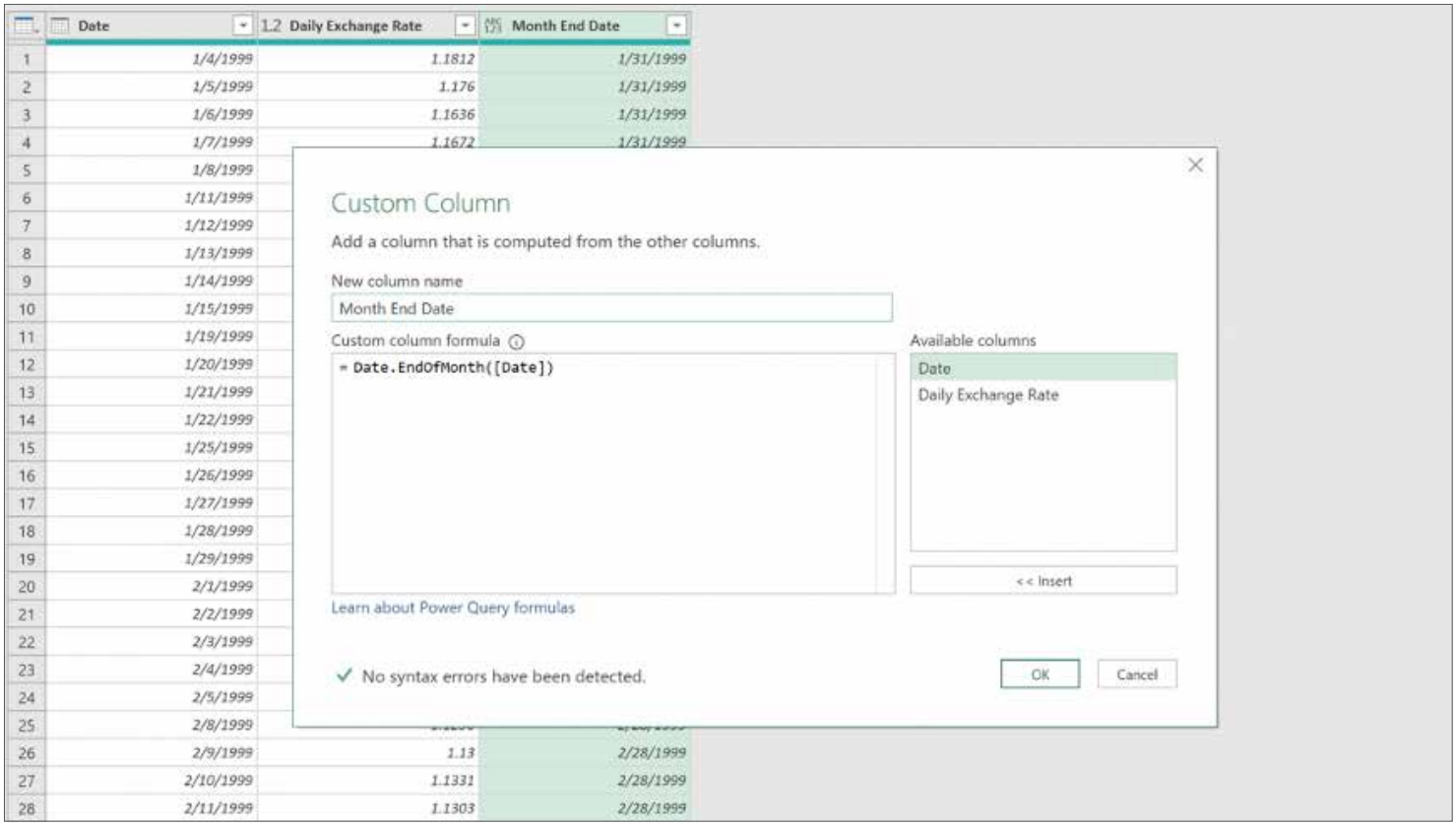


Figure 8: Month End column

the grouping functionality, you first create a new column for the last day of the month for each existing date. Select to add a new column to the data table, then enter the formula to reference the existing date field you see in **Figure 8**.

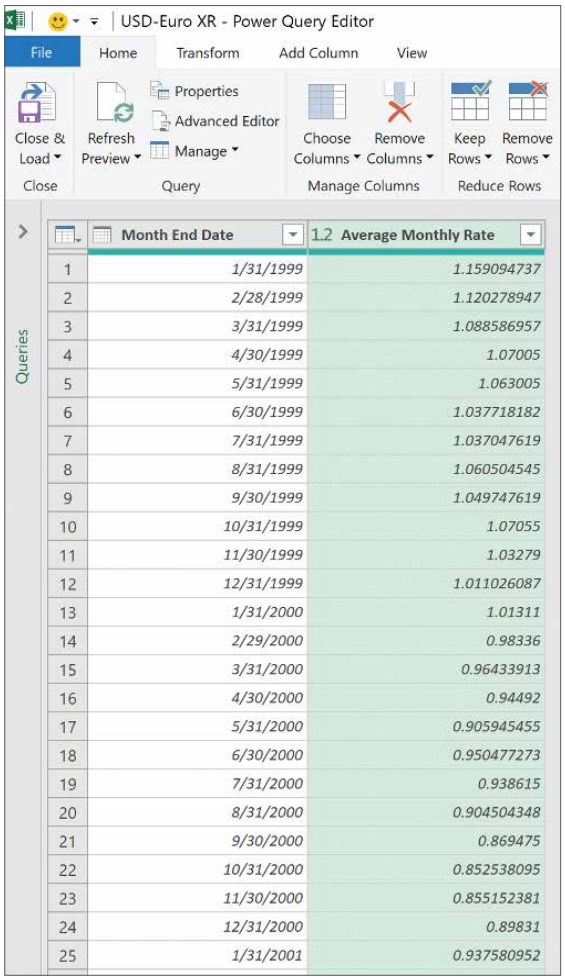


Figure 9: Returned table from grouping

Now you can group the data with the new date field. Select the Group By functionality from the top Transformation menu. In the dialog box group, use the Month End Date as the dimension in the dropdown menu, then create a new column named Average Exchange Rate by choosing the Average Operation from the dropdown menu to calculate using the Column of Daily Exchange Rate. In setting up this simple grouping, the query editor returns a table in a consolidated shape with two columns: the last day of the month column and the new average currency rate for the entire month (**Figure 9**).

You notice that this data grouping loses much of the initial data granularity. What if you wanted to calculate the average monthly exchange rate, but keep the daily exchange rates as part of the data table? To achieve this desired outcome, you will need to leverage the capabilities supported in the advanced grouping options.

Notice that the Grouping applied step in the query editor has a gear wheel icon next to the step name. Double clicking on this icon takes you back into the grouping configuration. This time, you want to select the advanced options, which enables you to set up more complex grouping configurations. Keep the date field from the existing configurations, as well as the aggregation for the average exchange rate. Since you want to return both the average monthly rates and daily exchange rates in the same view, you want to add a table object to each month that contain the daily rates. To do so, select to group All Rows, name this field Data, and notice that it doesn't allow you to choose an aggregation because you're returning all the rows (**Figure 10**).

After confirming this selection, you'll notice that the resulting data table (**Figure 11**) looks like the monthly average rates, except you now see that the last column contains rows consisting entirely of table objects associated with each month's end date. Think of this as a table of data attached to each of the month end dates and their respective monthly average rates. When you expand out this new Data column containing table objects, you combine the two columns of monthly aggregated data with each of the rows and columns contained in the table objects connected to the same date range, which gives you two rates for each monthly date.

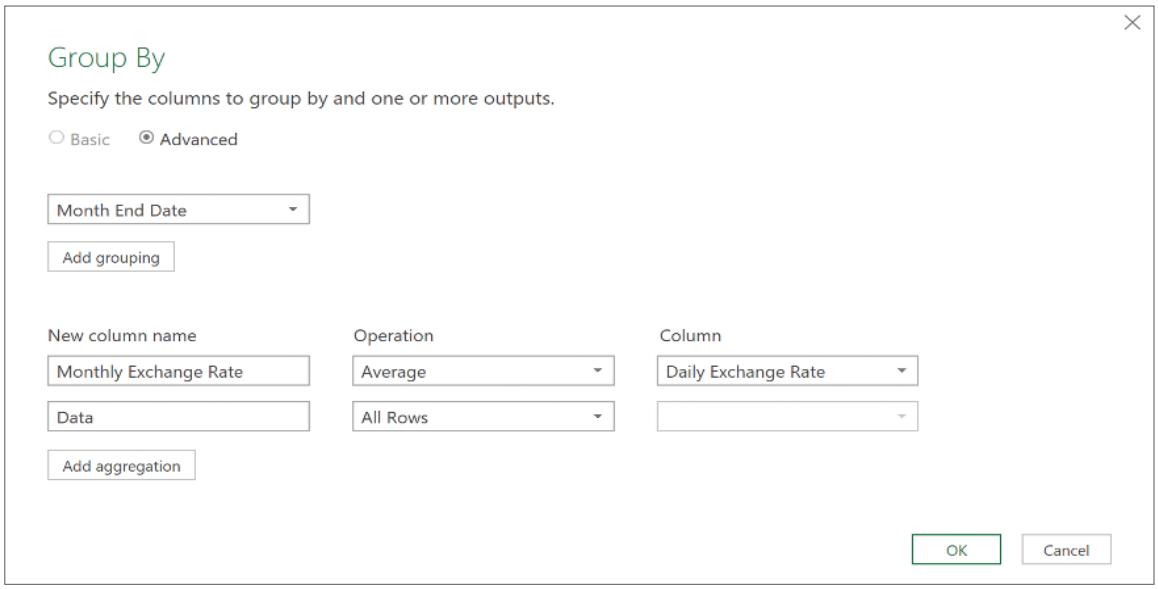


Figure 10: Advanced grouping configuration

By selecting the diverging arrows icon at the top of this new field, this expands to give you two rates for each date: the daily rate, and the monthly rate.

As an added challenge, you can add another calculation as a new grouping, this time for the year's end. You need to first create a new field for the year-end date using the same formula as **Figure 8**, except changing the Month piece to Year in the Date function. You then add another grouping to the transformation steps that uses this year field as the grouping dimension, calculates the yearly average rate instead of the monthly average rate, and finally creates new table objects for each row that contain both the average monthly rates and daily rates (**Figure 12**). You then expand out the Data field for the new grouped table.

	Month End Date	1.2 Monthly Exchange Rate	Data
1	1/31/1999	1.159094737	Table
2	2/28/1999	1.120278947	Table
3	3/31/1999	1.088586957	Table
4	4/30/1999	1.07005	Table
5	5/31/1999	1.063005	Table
6	6/30/1999	1.037718182	Table
7	7/31/1999	1.037047619	Table
8	8/31/1999	1.060504545	Table
9	9/30/1999	1.049747619	Table
10	10/31/1999	1.07055	Table
11	11/30/1999	1.03279	Table
12	12/31/1999	1.011026087	Table
13	1/31/2000	1.01311	Table
14	2/29/2000	0.98336	Table
15	3/31/2000	0.96433913	Table
16	4/30/2000	0.94492	Table
17	5/31/2000	0.905945455	Table
18	6/30/2000	0.950477273	Table
19	7/31/2000	0.938615	Table
20	8/31/2000	0.904504348	Table
21	9/30/2000	0.869475	Table
22	10/31/2000	0.852538095	Table

Figure 11: Returned table from advanced grouping

Step 4: Load Data to Excel

After creating this data table, you can remove all the date fields except the daily date (**Figure 13**. Lastly, you load the data to Excel (you can also load this same query into Power BI). Although you can theoretically load just over a million rows to Excel, this wouldn't be ideal because loading that much data severely impacts performance and fewer than 60,000 live calculations.

The loaded data appears as a separate tab within the Excel file that you can then rename to something like "US-Euro XR." You can also delete the other unneeded tabs in the Excel file, if you'd like. You can see that this data set is small, but if you're loading large data sets, you can select to only load the connections. Although this means you now can't see all the data in the same views, you can still build impactful data summaries such as pivot tables within the file.

Step 5: Enable the Users to Refresh Data

Finally, you want to make your Power Query work accessible to others. You don't want to update this file yourself every month, especially if you can avoid it. How, then, can you configure it as a user-centered process with the end business user in mind? You passed your FRED credentials into Power Query with the username and password. By sharing your file with others, they won't see your private log-in information, but they can refresh the query on their own computer.

To refresh the queries, you can either go back into Power Query to manually refresh the data, or you can manually refresh it with the refresh buttons within the Excel file. Although you or I may view this as the perfect bridge between the Excel view and the data in the API query, remember that business users may not have the same comfort with updating queries this way.

You can use a VBA macro to create a refresh button that the user can select to automatically update the queries directly in Excel.

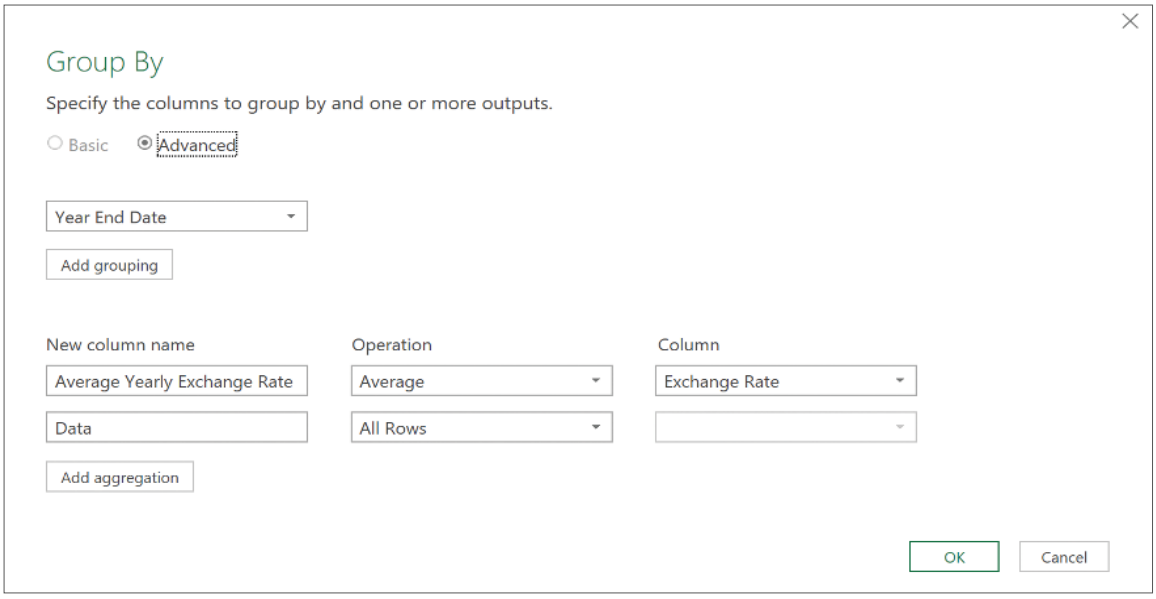
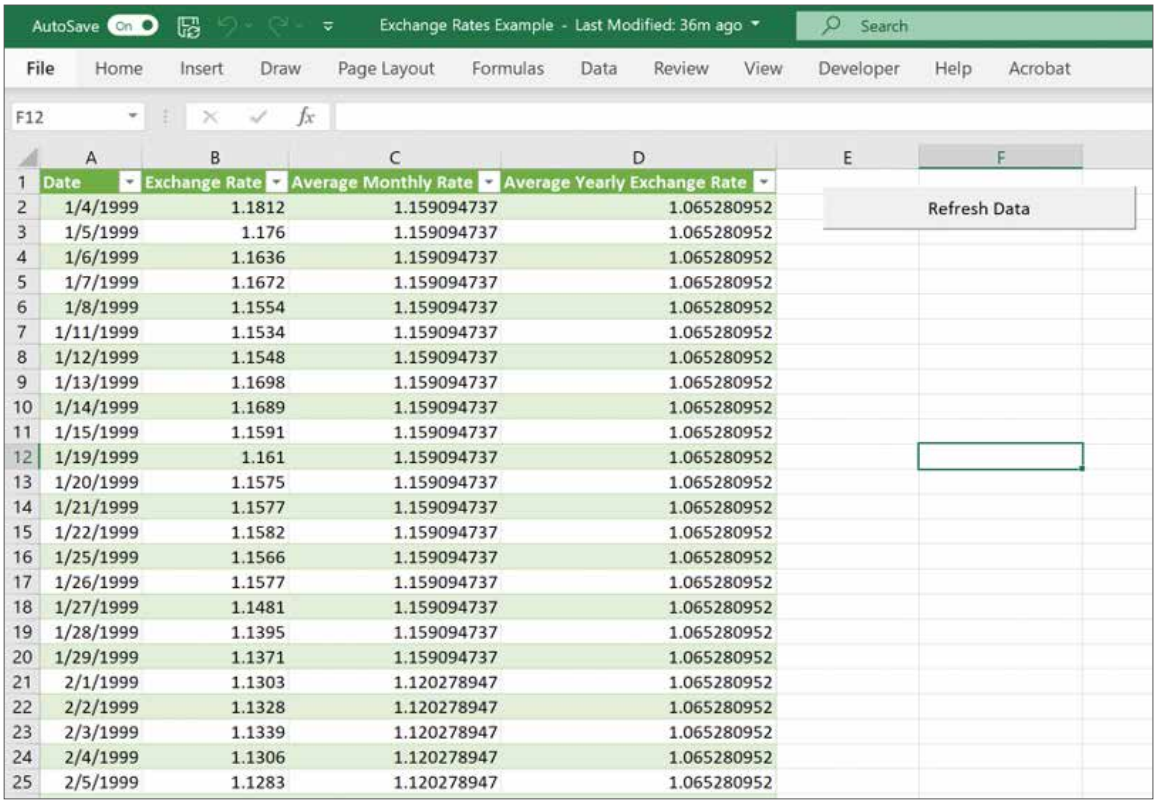


Figure 12: New grouping by year-end date



	A	B	C	D	E	F
1	Date	Exchange Rate	Average Monthly Rate	Average Yearly Exchange Rate		
2	1/4/1999	1.1812	1.159094737	1.065280952		
3	1/5/1999	1.176	1.159094737	1.065280952		
4	1/6/1999	1.1636	1.159094737	1.065280952		
5	1/7/1999	1.1672	1.159094737	1.065280952		
6	1/8/1999	1.1554	1.159094737	1.065280952		
7	1/11/1999	1.1534	1.159094737	1.065280952		
8	1/12/1999	1.1548	1.159094737	1.065280952		
9	1/13/1999	1.1698	1.159094737	1.065280952		
10	1/14/1999	1.1689	1.159094737	1.065280952		
11	1/15/1999	1.1591	1.159094737	1.065280952		
12	1/19/1999	1.161	1.159094737	1.065280952		
13	1/20/1999	1.1575	1.159094737	1.065280952		
14	1/21/1999	1.1577	1.159094737	1.065280952		
15	1/22/1999	1.1582	1.159094737	1.065280952		
16	1/25/1999	1.1566	1.159094737	1.065280952		
17	1/26/1999	1.1577	1.159094737	1.065280952		
18	1/27/1999	1.1481	1.159094737	1.065280952		
19	1/28/1999	1.1395	1.159094737	1.065280952		
20	1/29/1999	1.1371	1.159094737	1.065280952		
21	2/1/1999	1.1303	1.120278947	1.065280952		
22	2/2/1999	1.1328	1.120278947	1.065280952		
23	2/3/1999	1.1339	1.120278947	1.065280952		
24	2/4/1999	1.1306	1.120278947	1.065280952		
25	2/5/1999	1.1283	1.120278947	1.065280952		

Figure 13: Finished refreshable Excel file

I'll be the first to admit I'm not a huge fan of VBA. One of my pain points with VBA is that it still doesn't mitigate the risk of making easy mistakes that are difficult to correct. I do, however, like adding a button to the screen if it means that the business users will continue to reference my work in the long run.

This is done in two parts: First create the macro, then add a button to automatically refresh the data. Of course, you'll find a few more specifics in the process of setting it up, but you get the idea. Start from the Excel file view with a single sheet of the exchange rate data.

1. In the top ribbon, go to the Developer tab and open the Macros menu on the left.
2. In the VBA window now open in the view, select your file name, which should have the prefix VBA. Right-click and select Insert Module from the drop-down menu.
3. A new window opens where you can copy and paste the code below into the window to enable you to run a macro to update the Power Query data. Closing out the window saves the VBA code.

```
Public Sub Refresh()
' Macro to update my Power Query script(s)
Dim lTest As Long, cn As WorkbookConnection
On Error Resume Next
For Each cn In ThisWorkbook.Connections
lTest = InStr(1, cn.OLEDBConnection.Connection,
"Provider=Microsoft.Mashup.OleDb.1", vbTextCompare)
If Err.Number <> 0 Then
Err.Clear
Exit For
End If
If lTest > 0 Then cn.Refresh
Next cn
End Sub
```

Once you set up the macro to update the Power Query data, you can now turn your focus to adding the elusive update button next to the data, easily within view.

1. Select the Insert drop-down menu from the Developer tab of the ribbon and select the rectangle shape from this drop-down menu.
2. Drag this shape to the data table sheet where you want to drop it. I put the macro update button to the right of the data table to make it easy for the user to see immediately when they open the Excel file.
3. When you let go, Excel prompts you to assign a macro to this button, so select UpdatePowerQueries.
4. Select the text on the button to rename it something meaningful, like "Refresh Data". To double check that the button works, hit the button and make sure that it now includes any new records. Of course, if you create this entire project in the span of a single day, you won't see any new records on the same day, but you can return to the file the next day to check.

There you have it! In **Figure 13**, you have a smartly built, yet simple to set up and understand business solution that makes everyone's life easier. If you're interested in learning more about how Power Query works, you can check out my Power BI Data Methods course in the LinkedIn Learning library. Yes, it's a subscription service, but it does offer a lot to the customers!

Helen Wall

Transformation Options

The transformation options in Power Query are far more numerous than simply removing and renaming fields. There are three categories of transformation functions.

Cleaning data includes removing unneeded spaces and changing capitalization options. Shaping data includes grouping or unpivoting the entire table into a new table shape. Adding new fields includes standard mathematical functions, but also a plethora of unique Power Query formulas, such as dates and reading data, from different file types.

Vue 3: The Changes

In the last couple of years, the Vue framework has etched its place into the heart of many a Web developer. The team has been working on some major improvements the last couple of years that culminates in a beta of Vue 3 that has recently started shipping. In this article, I'll walk you through the pertinent changes that are coming to your beloved Vue. Although many of the changes are in the underlying plumbing, there are some major changes that I'll detail in this article. The overarching changes include:

- Conversion to TypeScript to improve type inference for TypeScript users
- Switch to using the Virtual DOM for overall performance improvements
- Improve tree-shaking for Vue users using WebPack (or similar solutions)

Overall, you should notice a much-improved experience without sacrificing compatibility with Vue 2. Although there are breaking changes, they've been kept to a minimum and allow new features to be opted into instead of forced upon existing code. Let's get to the changes!

Shawn Wildermuth
shawn@wildermuth.com
wildermuth.com
twitter.com/shawnwildermut

Shawn Wildermuth has been tinkering with computers and software since he got a Vic-20 back in the early '80s. As a Microsoft MVP since 2003, he's also involved with Microsoft as an ASP.NET Insider and ClientDev Insider. He's the author of over twenty Pluralsight courses, written eight books, an international conference speaker, and one of the Wilder Minds. You can reach him at his blog at http://wildermuth.com. He's also making his first, feature-length documentary about software developers today called "Hello World: The Film." You can see more about it at http://helloworldfilm.com.

The Current State of Vue 3

As of the writing this article, Vue 3 is in an early beta cycle. The cadence of betas is pretty quick. You can build from the source to get the absolutely latest version of Vue, but in any case, it's likely not time to start building or converting your projects. Some parts of Vue 3 (especially the big changes) are available as plug-ins to Vue 2, in case you want to start experimenting with them (I'll mention them in the article below as I talk about specific features). The team's goals are to make it highly compatible with Vue 2 so that you shouldn't need to change code to move to the new version, but instead opt into new features.

Although the core library of Vue is now in a beta of version 3, that's not true for the entire ecosystem. Most of these libraries are in alpha or preview states. From public commitments from the team, these libraries should be at version 3 by the time the core library is released. These include:

- vue-router
- vuex
- eslint-plugin-vue
- vue-test-utils
- vue-devtools

Additionally, the Vue CLI will be updated to use the Vue 3 libraries once it ships. There is currently a CLI plug-in for upgrading your projects to Vue 3 to see how they work. This add-in is only for experimental use, as of the writing of the article. To use it, just open an existing Vue CLI project and type:

```
# in an existing Vue CLI project
vue add vue-next
```

This will upgrade the project to the latest beta versions.

To look at the source or get the latest versions for all things related to Vue 3, please visit GitHub at: https://github.com/vuejs/vue-next.

New Features

In the years since Vue 2, the landscape of client-side Web development has changed quite a bit. Coming from a small upstart to a fully-fledged SPA library with a lot of community support, Vue has really grown up. Now with version 3 on the horizon, the team wanted to support a number of features to augment the library, simplify coding on Vue, and adopt modern techniques of Web development.

Below, I've dug into the major new features, but this list isn't exhaustive. It should cover the big items that will impact how you develop Vue applications going forward.

Global Mounting

One of the first changes you'll see is that mounting your root Vue object should be clearer than it was in version 2. The Vue object now supports a **createApp** function that allows you to mount it directly onto an element.

```
Vue.createApp(App)
   .mount("#theApp");
```

The fluent syntax creates an application by passing in a component and allows you to mount it to a specific element (via a CSS Selector). This doesn't change how the applications are created, but it does make it clearer what is happening. In Version 2, there were just too many ways to kick off your project; I really like this change.

In this same way, createApp returns an object in which to do configuration at the app level (instead of version 2's global-only option). For example:

```
createApp(App)
  .mixin(...)
  .use(...)
  .component(...)
  .mount('#app')
```

Back in version 2, you'd add mix-ins, use plug-ins, and add components (etc.) to the global Vue object. This allows you to scope them to your application and should cause fewer conflicts between libraries.

Composition API

This feature is the one that caused the most controversy with the Vue community. The Composition API is simply a new way of developing components that's more obvious but is a stark change to the Options API (the Vue 2 default way to build a component).

The motivation behind the Composition API is to improve the quality of the code. It does this by allowing you to decouple features and improves the sharing of that logic. In Vue 2, developers were forced to rely on extending the **this** object that was passed around to extend and share logic. This approach caused problems in that it was more difficult to see features

as they were added. This was especially exacerbated when using TypeScript (e.g., lack of typing). The upgrade also allows you to more obviously share features via standard JavaScript/TypeScript patterns instead of inventing new ones.

As another benefit of the Composition API is that types are easier to infer, which means better support for TypeScript. That was a big motivation for version 3. Let's see how it works in practice.

The Composition API works by changing from an options object to a composing of a Vue object. For example, the default Vue 2 way to create a component looks like this:

```
export default {
  data: () => {
    return {
      name: "Shawn"
    }
  },
  methods: {
    addCharacter() {
      this.name = this.name + "-";
    }
  },
  computed: {
    allCaps() {
      return this.name.value.toUpperCase();
    }
  },
  onMounted() {
    console.log("Mounted");
  }
}
```

In contrast, the Composition API, simplifies that all into a method called 'setup' that has you compose the same thing:

```
export default {

  setup() {
    const name = ref("Shawn");
    const allCaps = computed(() => name.value.toUpperCase());
    const address = reactive({
      address: "123 Main Street",
      city: "Atlanta",
      state: "GA",
      postalCode: "12345"
    });
    const favoriteColors = reactive(["blue", "red"]);
    function addCharacter() {
        this.name = this.name + "-";
    }

    onMounted(() => {
      console.log("Mounted");
    });

    return {
      name,
      address,
      favoriteColors,
      allCaps,
      addCharacter
    }
  },
}
```

The difference here is that you're generating an object that you return with the interface for the component. Because it's all uses the same scope, you can easily use closures to more easily share data instead of depending on the magic of the **this** property (the computed value is just getting the name via a closure). This should also allow you to decompose a component into several files, if necessary, and to manage large components, which was difficult with the Options API.

Reactivity

The way that reactivity worked in version 2 of Vue wasn't exposed to the developer. It was trying to hide the details, which caused more confusion than it should have. To remedy this, Vue 3 supports several ways of wrapping objects to ensure that they are reactive.

For scalar types, you can wrap them with a **ref** function. This makes them mutable and you read or write the property directly (if necessary) with **.value**:

```
const name = ref("Shawn");
const allCaps = computed(() => {
  name.value.toUpperCase()
});
```

For templates, objects that are wrapped as **ref** don't need to use **.value,** as those are unwrapped by default:

```
<template>
  <div>
    <label>Name: </label>
    <input v-model="name">
    <div>{{ name  }}</div>
    <div>{{ allCaps  }}</div>
    <button @click="addCharacter">Add</button>
  </div>
</template>
<script>
```

For objects, you would wrap it in a **reactive** wrapper. This wrapper makes a deep proxy for the object so that the entire object is reactive:

```
const address = reactive({
  address: "123 Main Street",
  city: "Atlanta",
  state: "GA",
  postalCode: "12345"
});
```

The object returned in the set-up function (from the Composition API) is automatically wrapped in a reactive object, so you don't need to do it unless you have your own reactive objects.

For arrays, reactive also works. You just need to wrap it in the same way inside of **setup**():

```
const favoriteColors =
  reactive(["blue", "red"]);
```

Wrapping arrays with reactive makes it more obvious when you modify the array. It may seem like this is a lot like RxJs because it is. My understanding is that you can opt to use RxJs, too. The reactivity is the requirement for how Vue 3 works. Although my example shows reactivity in a lot of

places, in most cases, libraries can hide some of these details (e.g., Vuex).

Composition Components

Along with the changes coming to the composition APIs, many of the utility components that were hung on the Vue object (e.g., $watch, computed) are now separate components to make that code more readable:

For example, if you import watch, you can then watch one or more refs or reactive properties:

```
import { ref, reactive, computed, watch }
  from "vue";

// Watch a scalar ref
watch(name, (name, prevName) => {
  console.log(`Name changed to: ${name}`);
});

// watch a property of a reactive object
watch(
  () => address.postalCode,
  (curr, prev) => {
    let msg = `Postal Code: ${prev} to ${curr}`
    return console.log(msg);
  }
);
```

Similarly, computed is now an importable component:

```
import { ref, reactive, computed, watch }
  from “vue”;
export default {
  setup() {
    const name = ref(“Shawn”);
    const allCaps = computed(() => {
      return name.value.toUpperCase();
    });
    return {
      name,
      allCaps
    }
  },
}
```

This should make the composition of your components a lot clearer and the code that much cleaner (instead of relying on a heavily overloaded **this** property.

Filters

One of the big surprises in Vue 3 is that the concept of filters is going away. There are some justifications for this but the main one is that Vue 3 wants the inside of a bindings to be simply executable JavaScript. Let me show you what I mean. Here is a simple filter usage in Vue 2:

```
<div>{{ name | uppercase }}</div>
```

This syntax was an exception to most of the ways that binding worked. This is because the syntax using the pipe (|) character was confusing and not valid JavaScript. The team decided that we should do the same thing without resorting to some special syntax. Because the binding can be any valid JavaScript, you can accomplish the same thing with computed properties or just simple functions. In this example, you can see that the uppercase filter is changed to a function that returns uppercase:

```
export default {
  setup() {
    const name = ref("Shawn");

    function uppercase(val) {
      return val.toUpperCase();
    }

    return {
      name,
      uppercase
    };
  }
};
```

In Vue 2, I’m used to registering global filters. Instead, I think an approach is to just create your filters as an importable object and then just use the spread operator to add them to the binding object. For example, here’s a small version of a library that could hold one or more filters:

```
// filters.js
export default {
  uppercase(val) {
    return val.toUpperCase();
  }
};
```

Then I could just import it and apply it:

```
import filters from “./filters”;

export default {
  setup() {
    const name = ref(“Shawn”);

    return {
      name,
      ...filters
    };
  }
};
```

Then you can just use it as a function in the binding scope:

```
<div>{{ uppercase(name) }}</div>
```

Because the binding scope is just JavaScript, if you need to specify additional parameters, you can. Go nuts with it. It’s more obvious and reduces complexity. After being scared at first, I like it.

Suspense

The Vue 3 team is making an effort to learn from other ecosystems. An example of this is the new support for Suspense (like React’s common pattern). The problem that Suspense is trying to solve is to allow you to specify some template to be shown in case your components need to do any asynchronous work (e.g., calling the server) before being ready. In this case, Suspense will do this for you.

Let’s start with an example. Say I have a simple component that has to do something; it makes its set-up use async and

await so it can handle the call. In this example, I'm doing a simple timeout:

```
<template>
  <div>
    <div>See this works now...</div>
  </div>
</template>
<script>
export default {
  name: "ShowSomething",
  async setup() {

    await new Promise(res => {
     return setTimeout(res, 2000);
    });

    return {

    };
  }
}
</script>
```

Then, on a parent component that uses this component, there normally wouldn't be any mechanism to know it's asynchronous. So you bring in the component to the parent component:

```
import ShowSomething
  from "./components/showSomething";

export default {
  setup() { ... },
  components: { ShowSomething }
};
```

Instead of writing a simple template that uses the component, you use a suspense block to specify two separate templates (marked **default** and **fallback**):

```
<template>
  <Suspense>
    <template #default>
      <div>
        <ShowSomething />
      </div>
    </template>
    <template #fallback>
      <div>
        <div>Please wait...</div>
      </div>
    </template>
  </Suspense>
</template>
```

In this case, you can see that what will happen is that until the **ShowSomething** component is done with its **setup**, the **fallback** template will be used. When it's complete, it switches to the **default** template. Neat!

Teleport

Another interesting feature that started its life in React is the idea of Teleport (or Portals as React calls it). The basic idea of teleport is to be able to render some part of a component in a DOM element that exists outside the current scope. For example, you can add a Teleport element with some content that you want to render:

```
<div>
  <Teleport to="#appTitle">
    <h3>{{ name }}</h3>
  </Teleport>
...
```

And on your webpage, you might have a div with the ID of appTitle:

```
<p>
  Teleport Should Appear here,
  not inside the component:
</p>
<div id="appTitle"></div>
```

When rendered, the contents will be "teleported" to the appTitle div and shown in that part of the UI—it doesn't need to be inside your application at all. Ta-da!

Routing

Although Vue Routing has been upgraded to Vue 3, there were some changes to the API to make it more consistent—overall, the goal was to make sure that it was more compatible with TypeScript. Routing is, on the whole, backward compatible with Vue Routing version 3 (which was used in Vue 2). There are some breaking changes, but they are relatively minor including:

- Specifying the routing mode is now done via a new history property instead of a mode property
- The base of routing is now specified when you specify history
- Catch-all routes have changed format
- Transitions must now wait until the router is ready

You can review the breaking changes on the repository: https://github.com/vuejs/vue-router-next.

Vuex

The overall API for Vuex has been kept for this new version of Vuex for Vue 3. But some of the ways you wire-up the store are different. For example, although you can still create a Vuex Store by calling **new Vuex.Store**, they're suggesting that you use the **createStore** exported function to better align with the way that Vue Router. So, in version 2, you'd create a new store like this:

```
import Vuex from 'vuex'

export default new Vuex.Store({
  state: {
  },
  mutations: {
  },
  actions: {
  },
  modules: {
  }
});
```

In contrast, in version 3, they suggest that you create it like so:

```
import { createStore } from 'vuex'

export default createStore({
  state: {
  },
  mutations: {
  },
  actions: {
  },
  modules: {
  }
});
```

With the changes to global mounting, this changes how you'd wire up the store as well:

```
import { createApp } from 'vue';
import App from './App.vue'
import store from './store'

createApp(App)
  .use(store)
  .mount('#app')
```

Except for these minor changes, your Vuex code should just continue to work.

Where Are We?

Let me be honest. I'm sure that since we're not at release yet, I'm missing a couple of breaking changes. The purpose of this article isn't to be completely exhaustive but instead to prepare you for the major changes coming to Vue. On the whole, I really like the new changes and think they are a definite improvement in the library. If you don't agree, don't worry. You can stay with Vue 2 as long as you like. They are back-porting some of the major changes to allow you to continue using Vue 2 if you like, although the back-porting will happen sometime after Vue 3 releases.

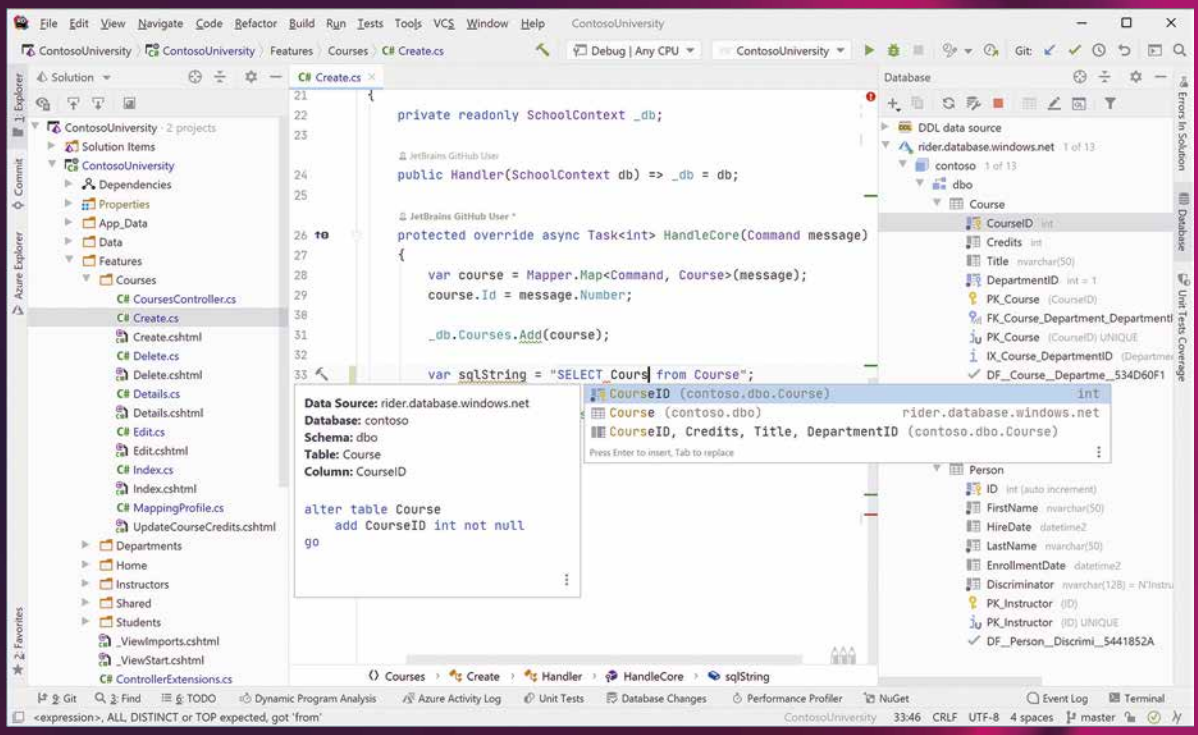
I hope this article has helped gird you for the upcoming changes and, at best, gets you excited to try the new version of Vue.

Shawn Wildermuth

Rider

JET BRAINS

Fast & powerful, cross-platform .NET IDE



One IDE for everything
Write code using C#, ASP.NET Razor, JavaScript, TypeScript, HTML, CSS, and JSON.

Support for various web frameworks
Use Node.js, React, Angular, or Vue.js in your project.

Database tools
Connect to databases, run queries, and enjoy schema-aware code completion for MSSQL, MySQL, PostgreSQL, and other systems.

Productive developing experience
Take advantage of fast code analysis, code completion, refactorings, debugging, and unit testing capabilities.

Cloud integration
Deploy web apps to Azure or AWS.

jb.gg/tryrider

Stages of Data: A Playbook for Analytic Reporting Using COVID-19 Data

Over the last few years, I've written articles about ETL and back-end processes in data warehousing applications. A few months ago, I started writing a similar article on a "playbook" for analytic reporting. I've built many business intelligence applications for several industries. Although the specifics differ, there are common key elements. Just like Visual Studio and .NET have properties, events, and methods, BI applications have flashpoints, activities, and strategies that occur along the life cycle of the application. I began to write an article that organized these "PEMs" into a playbook. I initially took an actual business application and mocked up the data for the example. Then in February/March of 2020, news of COVID-19 began to affect us all. I changed my example application to use COVID-19 data and will present it here.

Kevin S. Goff
kgoff@kevinsgoff. net
www. KevinSGoff. net
@StagesOfData

Kevin S. Goff is Database architect/developer/speaker/author, and has been writing for CODE Magazine since 2004. He was a member of the Microsoft MVP program from 2005 through 2019, when he spoke frequently community events in the Mid-Atlantic region and also spoke regularly for the VS Live/Live 360 Conference brand from 2012 through 2015.

Disclaimer

Before I talk about my goals for this article, I realize that writing about COVID-19 and presenting a reporting application can seem odd and even morbid. Every number is a human being, and as I type this, the United States and World Death counts are roughly 87K and 312K respectively and growing every day. These are victims with family and friends.

We have health care workers who are dealing directly with the crisis of their lifetimes. Millions of individuals have lost their jobs and/or have experienced major economic hardships. Parents and children are dealing with the massive shift in the education model and the challenges of kids staying at home. For all the items I just mentioned, there are countless others. I struggle to find the words to convey how serious this is.

Over the years, I've collected data for personal projects, ranging from my daily health numbers to election counts to sports statistics. I started collecting data on COVID because I wanted to track the trends—not just in the big cities, but in smaller areas where I'd heard news stories of outbreaks. Along the way, several people alerted me to some very attractive public dashboards on COVID data. They were visually impressive, but I found that several lacked the types of analytic paths I wanted to track. (I'll cover those below in the "Goals for This Application" section).

We see coverage of these numbers every day, with many positions/opinions related to policy decisions to deal with this crisis. I'm writing this without injecting any personal view or bias. Because I can never resist a pop culture analogy, many will recall that in all the years of the Johnny Carson "Tonight Show," he talked about all sorts of news events, but you never knew his opinions on the content. Likewise, I'm not offering any opinion or looking to make any statement about potential policy decisions to deal with COVID, other than to state the obvious: There are no easy answers for this.

In this article, I'm going to use some of the mapping features in Power BI, and to keep things simple, I'm restricting the scope of this article and COVID data to the "lower-48 states" in the United States. I acknowledge that CODE has many readers outside the U.S, as well as the U.S. states of Alaska and Hawai'i. I'm in no way minimizing their shared experiences through this horrible crisis. Writing an article and presenting a case application requires some economizing in a fixed window of time, and for that reason I've restricted this to the 48 states.

Also, many in the news have discussed the accuracy of the case/death counts and the context surrounding the data collection. I'm going to assume, for purposes of this article, that the numbers are valid. At the end of the article, I've listed all of my data sources.

I understand if readers react that this is too dark a topic. But to the degree that people compare current events to past history of pandemics, discussions today potentially become the history of tomorrow. The books we read today on pandemics from 50-100 years ago (filled with statistics) were at one time "current events".

Goals for This Application

I've looked at several public COVID-19 dashboard reporting applications. Some have eye-popping displays, but don't always have as many drill-down features. Some show more detailed information but spread across several pages in such a way that putting it together to form a picture is difficult.

Building the proverbial Holy Grail of reporting applications takes a long cycle of continuous improvements. There's an old variation of a Murphy's Law joke that the perfect application does not exist, and if it did, few would likely use it. Even so, I wanted to put together a reporting piece that would cover some of my questions about numbers across the country with respect to certain metrics, such as population densities. Therefore, I decided to re-work my article to cover a COVID data and reporting application. I'll present the application and then talk about the "plays" in my report application playbook to see how many plays I actually covered.

First, the granularity of the data would be daily cases count and deaths by United States County.

- The county population (from 2019)
- County square miles
- Population per square mile
- Daily count of new CV cases and CV deaths
- CV deaths as a % of cases
- CV cases and CV deaths as a percent of population, square mile, and population/square mile

Here are some of the things I'd like to include:

- Summaries of this information by week, because there can be dramatic day-to-day changes

- Ranking the counties and states in the U.S. by any of the listed metrics. This will allow me to find hotspots that some might not initially believe would be a hotspot.
- Assign a key performance indicator (KPI), to see which counties are trending downward enough to start re-opening. For instance, a state might declare that a county can re-open when the number of cases over the last two weeks is less than X cases per Y-thousand people
- Rendering this data in bar charts, tables, and basic state/county maps filled based on a metric.
- Easy drill-down from a state into the counties.
- When looking at cumulative numbers by county, easily see the daily and weekly progression of these metrics.

If you've been following the news on this general subject, you know that there's been discussion on hospital counts, hospital resource usage, nursing home statistics, and breakouts by age group. I wanted to include statistics in these areas, but for the following reasons couldn't:

- I tried to pull statistics on hospital case counts. I found some at local levels but not across enough of the country.
- I also tried to pull statistics on counts in long-term care facilities, as well as breakouts by age group. Confidentiality laws and regulations make this data hard to attain. As of early May 2020, roughly 35 of the 50 states provide some/all of these statistics. Currently the NY Times reports that as many as 33% of COVID-19 deaths in the United States are in nursing homes, with several states well above 50%. My home state of Pennsylvania reports nearly 70% of COVID deaths are in nursing homes, with some counties reporting as high as 80%.

Based on the Data, Find Out the Following

There are nine questions I'd like to use this application to answer. I've put them into three categories:

General statistics:

- On any given day, how did the states rank for a particular metric?
- For the state with the highest deaths, what was the county breakout?
- For a specific county and a rate/based metric (deaths per square mile, deaths per population), how easily can I see, all at once, the rate for the county, the related state, and the U.S.? (Note: at the risk of giving the plays away too soon, this is an important topic: the ability to show percentages for an entity, the parent entity, the grandparent entity, etc.)
- Even if I'm looking at cumulative numbers as of a certain point in time, can I see the daily/weekly progression?
- Can I sort counties across the country by one number (population, or population per square mile), and then compare another metric (deaths or cases per population)?

Mapping questions:

- Can I render this information in maps and show a geographic county map to illustrate which counties in a state have the worst metrics?
- I know that major cities like NYC have been devastated. However, there are smaller counties in the continental U.S. with very high death/population rates. Is there any easy way to see them?

State comparison question:

- We know that New York has had the highest state numbers. What about the next five highest states? How have they trended over the last month? And when I look at their county populations, is there direct correlation?

Assessment of readiness (Key Performance Index) question:

- As of this writing (early May 2020), some states have set guidelines for partial re-openings by county. For instance, in Pennsylvania, one major guideline is that, for the last 14 days, a county must have less than 50 total new cases in those fourteen days per 100,000 residents. For counties in PA (and for that matter, across the nation), which counties are in the best relative position to meet this, and which are the worst?

The Tools I've Chosen

I'm using SQL Server Integration Services to pull down daily CSV case/death counts. I used SQL Server to hold the data and Microsoft Power BI for the visualizations. Could I have done this in SQL Server Reporting Services? Yes. However, Power BI's dashboard offerings have improved to the point where it's a legitimate tool for this type of work. Yes, there are still SSRS features that should be in Power BI, but Power BI is still a strong tool for this type of work.

I deployed this report to my Power BI Pro site in the cloud, and I published a public version. My website (www.kevinsgoff.net) has a link to the application (the URL is much too long to include here and could change).

This article will contain some T-SQL code to deal with some allocation of data, and some Power BI DAX code to deal with dynamic measures/calculations. My website (www.kevinsgoff.net) has information on the public version of this Power BI application, along with the Power BI Desktop (PBIX) file and other necessary files.

Begin with the End in Mind: The Data

Here is the core data I need to collect. First, I need a table of counties in the continental United States, along with a recent population count and square mile measurement (**Figure 1**).

The population count comes from the July 1, 2019 United States Census Bureau. The FIPS code (Federal Information Processing Standard) is a combination of a zero-filled two-digit state code and a zero-filled three-digit county code. From **Figure 1**, 42045 uniquely indicates Delaware County in Pennsylvania. (There are six counties named "Delaware" in the United States). As it turns out, the COVID-19 daily count source I've been using has the FIPS code as the key.

In the Data Sources section at the end of this article, I've included the websites where I pulled the county FIPS codes

Population	StateName	CountyName	FIPSCode	SquareMiles
566,747	Pennsylvania	Delaware County	42045	184

Figure 1: County/State Master table with FIPS code, population, and square mileage

and population. I was determined to pull square miles by land, as I wanted to include population density as a metric. Given the early statistics on infection rates, population per square mile can be a major correlating factor.

This was quite a challenge: the only source I could find was a PDF that I had to convert to Excel, and then parse the columns carefully. The values had spaces instead of commas for any four-digit values, which made it tricky. Knowing this would likely be a one-and-done effort (county land area doesn't change much, and certainly not to any degree of analytic significance), I didn't mind doing a one-time pattern hack, so long as I could manually verify small and large counties in each state.

3/12/2020	Delaware	Pennsylvania	42045	1	0
3/13/2020	Delaware	Pennsylvania	42045	6	0
3/14/2020	Delaware	Pennsylvania	42045	6	0
3/15/2020	Delaware	Pennsylvania	42045	7	0
3/16/2020	Delaware	Pennsylvania	42045	7	0
3/17/2020	Delaware	Pennsylvania	42045	9	0
3/18/2020	Delaware	Pennsylvania	42045	14	0
3/19/2020	Delaware	Pennsylvania	42045	14	0
3/20/2020	Delaware	Pennsylvania	42045	23	0
3/21/2020	Delaware	Pennsylvania	42045	33	0
3/22/2020	Delaware	Pennsylvania	42045	43	0
3/23/2020	Delaware	Pennsylvania	42045	54	0
3/24/2020	Delaware	Pennsylvania	42045	84	0
3/25/2020	Delaware	Pennsylvania	42045	101	0
3/26/2020	Delaware	Pennsylvania	42045	156	1
3/27/2020	Delaware	Pennsylvania	42045	185	3
3/28/2020	Delaware	Pennsylvania	42045	226	4
3/29/2020	Delaware	Pennsylvania	42045	276	4
3/30/2020	Delaware	Pennsylvania	42045	303	4

Figure 2: CSV from daily web feed from GitHub (NY Times)

Next, I needed to find a daily count of confirmed cases and deaths by county. I found a great one from GitHub (listed in the Data Sources section at the end of this article). This daily feed is available for download as a CSV file and contains **cumulative** counts of cases and deaths by day/county (**Figure 2**).

As it turns out, this was all I needed. I wrote a basic SSIS package that parsed the columns into a table called **COVIDHistory** (**Figure 3**).

I want to report on daily numbers, even though the feed only contains cumulative counts for every county. Therefore, I need to write something to parse the difference in cases/deaths from one day going backwards. I want the result set like **Figure 4**:

> I was determined to pull square miles by land, as I wanted to include population density as a metric. Given the early statistics on infection rates, population per square mile can be a major correlating factor.

casedate	fipscode	cases	deaths
2020-05-01	42045	3895.0000	240.0000
2020-04-30	42045	3747.0000	235.0000
2020-04-29	42045	3672.0000	224.0000
2020-04-28	42045	3514.0000	164.0000
2020-04-27	42045	3405.0000	145.0000
2020-04-26	42045	3333.0000	143.0000
2020-04-25	42045	3213.0000	142.0000
2020-04-24	42045	3106.0000	131.0000
2020-04-23	42045	2963.0000	124.0000
2020-04-22	42045	2798.0000	124.0000

Figure 3: Cumulative counts stored into a table

fipscode	casedate	Cases	Deaths
42045	2020-05-01	148.0000	5.0000
42045	2020-04-30	75.0000	11.0000
42045	2020-04-29	158.0000	60.0000
42045	2020-04-28	109.0000	19.0000
42045	2020-04-27	72.0000	2.0000
42045	2020-04-26	120.0000	1.0000
42045	2020-04-25	107.0000	11.0000

Figure 4: Daily counts derived from cumulative counts

I can use a little bit of T-SQL, specifically the LAG function, to line up cases for the current day and the prior day.

Note: I am taking **significant advantage** of the fact that once a data feed identifies a county case for a day, the data feed continues to include that county on each subsequent data, even if the cumulative counts have not changed.

```
truncate table  covid19DailyCounts
insert into covid19DailyCounts
   (FipsCode, CaseDate, Cases, Deaths)
 select fipscode, casedate,
        cases - casesyesterday as Cases,
        deaths - deathsyesterday as Deaths
  from (select fipscode, casedate, cases, deaths,
    lag(cases,1,0) over
    (partition by fipscode order by casedate) as
          CasesYesterday,
    lag(deaths,1,0) over
     (partition by fipscode order by casedate) as
            DeathsYesterday
from CovidHistory ) temp
```

Finally, I want to create a daily table with the cumulative number of new deaths and cases in the last 14 days, relative to each day. (**Figure 5**).

At this point, I'm reminded of the old joke about getting 10 published economists in the room who give 10 different an-

casedate	fipscode	caseslast14days	DeathsLast14Days
2020-05-01	42045	1614.0000	166.0000
2020-04-30	42045	1692.0000	164.0000
2020-04-29	42045	1737.0000	164.0000
2020-04-28	42045	1663.0000	118.0000
2020-04-27	42045	1693.0000	105.0000
2020-04-26	42045	1739.0000	104.0000
2020-04-25	42045	1703.0000	103.0000
2020-04-24	42045	1729.0000	101.0000
2020-04-23	42045	1741.0000	98.0000
2020-04-22	42045	1764.0000	101.0000

Figure 5: Daily county record of cumulative cases and deaths over the last 14 days

swers to a question. (There's an alternate joke about getting 10+ answers!). Similarly, database people might look at my approach and say something like, "Why don't you just store those two columns in the snapshot history table and not create a new table?" or even, "Why are you materializing this data at all? Just let the reporting tool create it on the fly"

All valid questions. So long as I'm not talking about an astronomical amount of data, and so long as the ETL processes deal with this in an automated fashion, I don't mind materializing this into a separate table.

I've created a third table called CovidLast14days, using the CROSS APPLY function to marry up each county/date with the sum of cases and deaths going back the last 14 days inclusive. When Microsoft released the CROSS APPLY function, many writers (myself included) praised it as a way to apply (thus the name) the results of a table-valued function to a query. CROSS APPLY also has a nice ability to act as a COBOL-like "Perform Varying" to self-join a table based on a sliding condition (yes, I'm sure many readers just fainted that I used a COBOL analogy!)

```
truncate table Covid19Last14Days

insert into  Covid19Last14Days
   select casedate, fipscode,
     caseslast14Days, DeathsLast14Days
   from covid19DailyCounts outside
       cross apply
        (select sum(cases) as CasesLast14days,
                sum(deaths) as DeathsLast14Days
      from covid19DailyCounts inside
        where inside. fipscode =
                 outside. fipscode
              and inside. casedate between
            dateadd(d,-13,outside. casedate) and
                 outside. casedate) temp
```

As far as the data goes, that's it! **Table 1** contains the calculated measures I'd like to create in the Power BI report.

Some Source Data Challenges:

I encountered two data source challenges with the daily count feed. First, the data source summarized NY city as a whole, instead of breaking it out by the five counties/boroughs:

- New York County (Manhattan)
- Kings County (Brooklyn)
- Bronx County
- Richmond County (Staten Island)
- Queens County

I took the New York city tally as a whole and spread it across the five counties based on population. Yes, I could have

Variances over the Average

You might have thousands of customers and hundreds of product options. When comparing shipments and revenue and costs from the most recent quarter to the quarter before, there might have been three specific combinations that represented the single highest percentage of increase for a metric. Being able to bring this information to the surface EASILY can help the business to spot trends

Calculated Measure	Notes
Population across all 48 states	
Square miles across all 48 states	
Population Per Square Mile (current slice of data)	
Population Per Square Mile across all 48 states	
Cases Per Population	
Cases Per Population across all 48 states	
Cases Per Square Mile	
Deaths Per Case	
Deaths Per Population	
Deaths Per Square Mile	
CasesLast14DaysGoal	For current slice, the Population divided by 100K, then multiplied by 50. This is for the metric to compare the cumulative cases over the last 14 days.
CasesLast14DaysGoalPerformanceIndex	The performance index of the cumulative cases over the last 14 days against the goal. A county with a population of 200K would have a goal of no more than 100 new cases in the last 14 days. If they had 90 new cases in 14 days, the index would be. 9 (which is good). If they had 110 new cases in the last 14 days, the index would be 1. 1 (not as good).

Table 1: Calculated Measures

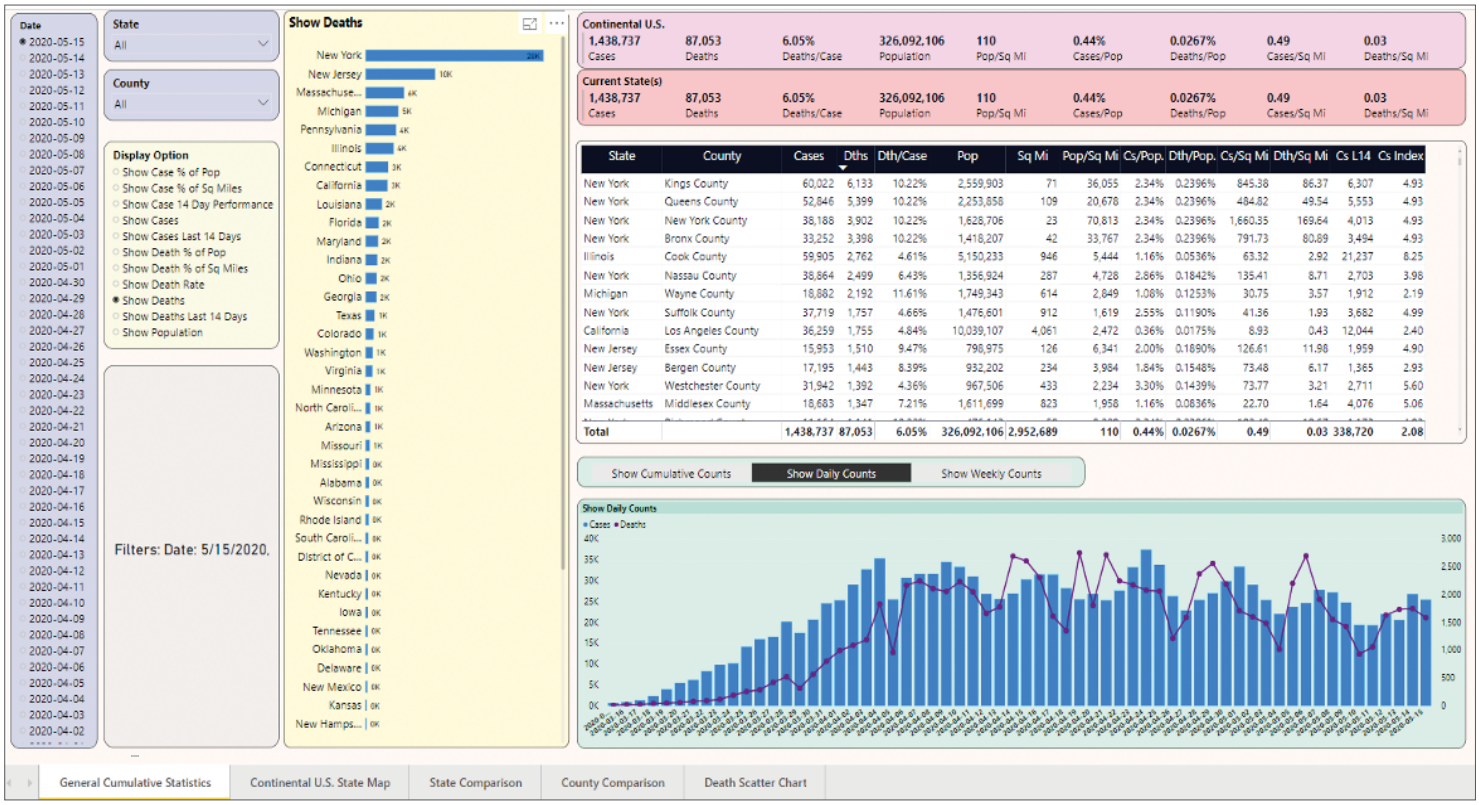


Figure 6: First page of general cumulative statistics and overall tabs

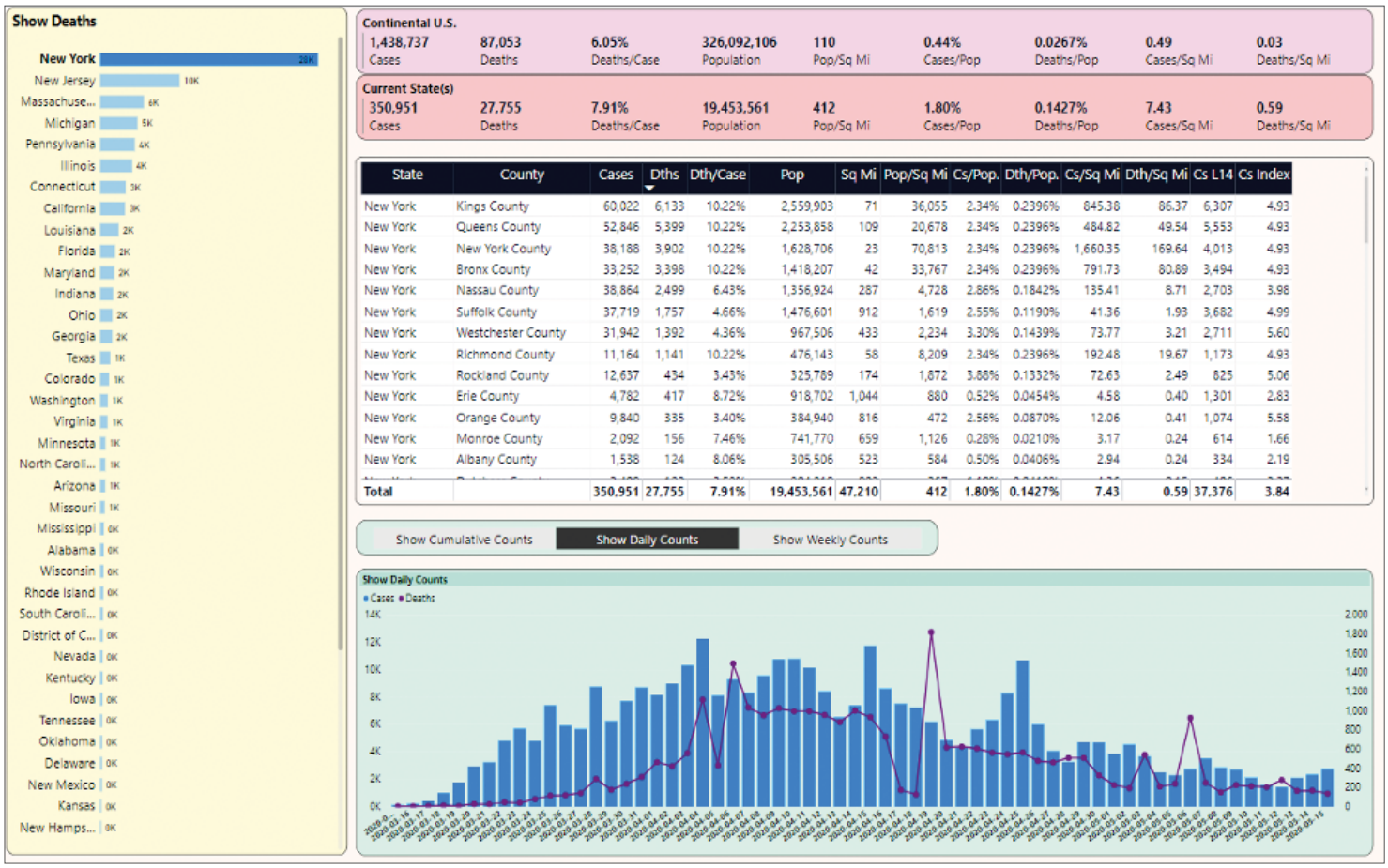


Figure 7: Comparing NY state measures to U.S. measures and ranking NY counties

searched for a second website source. Given the significant population density of these counties and the issue of county reporting, I can understand the challenges in trying to get a 100% accurate count down to the county level.

Second, the data feed contained some state tallies with a county of **Unknown**. To address this, I spread those counts across the counties in the state based on population.

The Power BI Application Pages

Here's the first page of the Power BI application (**Figure 6**), along with some tabs at the bottom for different pages. I'll address the questions above across the different pages.

General Statistics

Let's take the first five questions and walk through a few scenarios. I'll start with **Figure 7.** First, notice the drop-down for the Display Option. Based on that selection, the page ranks the 48 states based on that metric. I've selected "Deaths," although I could have picked any of the aggregated counts or any of the metrics from **Table 1**. (I'll talk about the drop-down for "Display Daily Chart Option" in a minute).

Note the filters caption, with the date of 5/15. I've selected 5/15 from a date list drop-down (from back in **Figure 6**, but not shown here, simply to conserve screen real estate). The output ranks the states by cumulative death counts, with NY first at 27,755, New Jersey a distant second at 10,138 deaths, and so on. Again, this is "as of 5/15". I could pick a prior day to view of snapshot of cumulative counts as of that date.

I've selected New York with the mouse, which filters the list of counties on the right. In that list of counties, I've sorted on the metric for Deaths Per Square Mile.

Note the chart in the lower right that currently shows daily case and death counts. If I want to show weekly counts instead (**Figure 8**), I can take advantage of a dynamic measure based on the user selection.

```
DynamicDailyDeathsMeasure =
   switch( values (  DailyTrendOption[Option]),
         "Show Daily Counts",
               sum(covid19DailyCounts[Deaths]),
         "Show Cumulative Counts",
               sum(CovidHistory[deaths]),
         "Show Weekly Counts",
               sum(covidweeklycounts[Deaths]))
```

With very little navigation, I'm able to see the following:

- The ranking of states by cumulative death count
- By selecting New York, I'm able to see (from the two pink rectangular panels in the upper right) that NY's Cases and Deaths Per Population and Per Square Mile are many times greater than the national average.
- The five counties that make up NYC (New York/Manhattan, Kings, Bronx, Queens, and Richmond) have

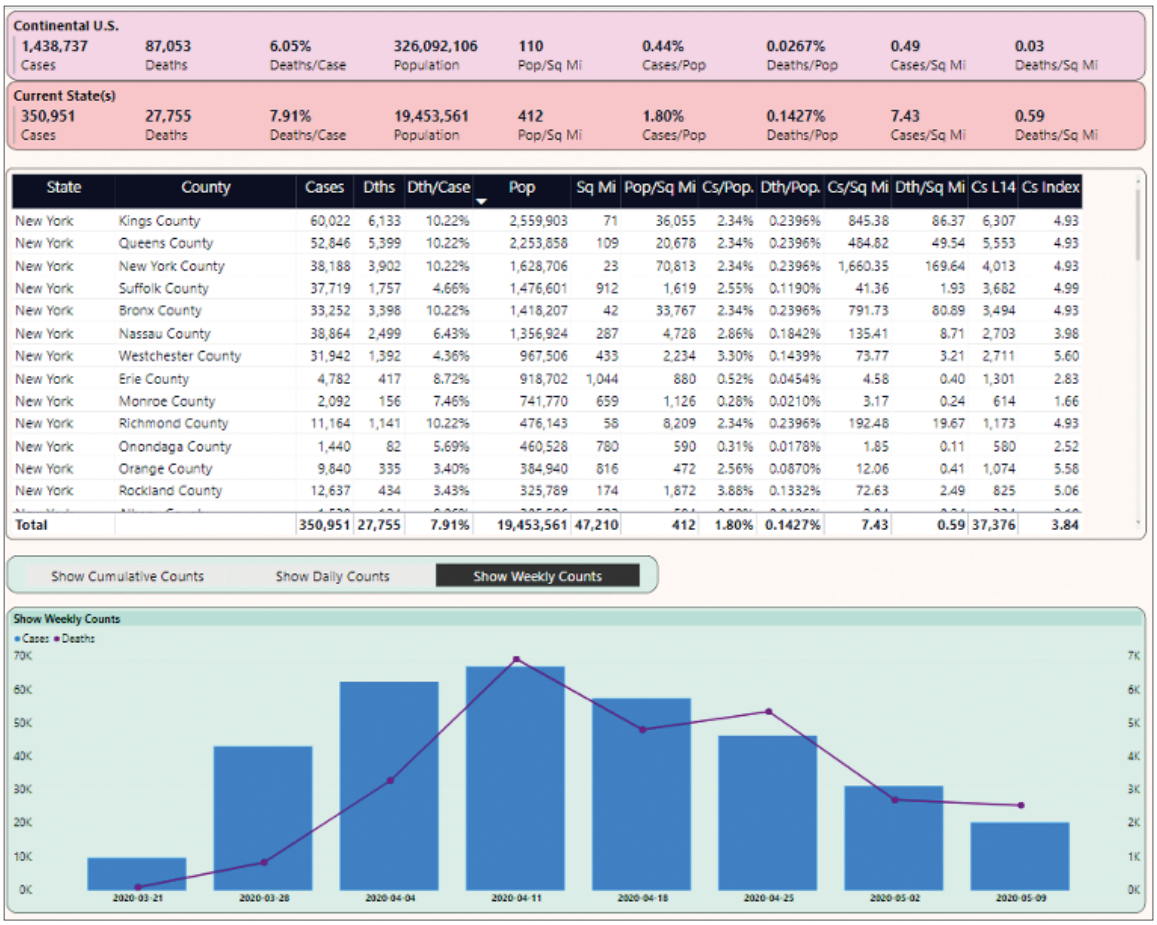


Figure 8: Same as Figure 7, but showing weekly counts instead of daily counts

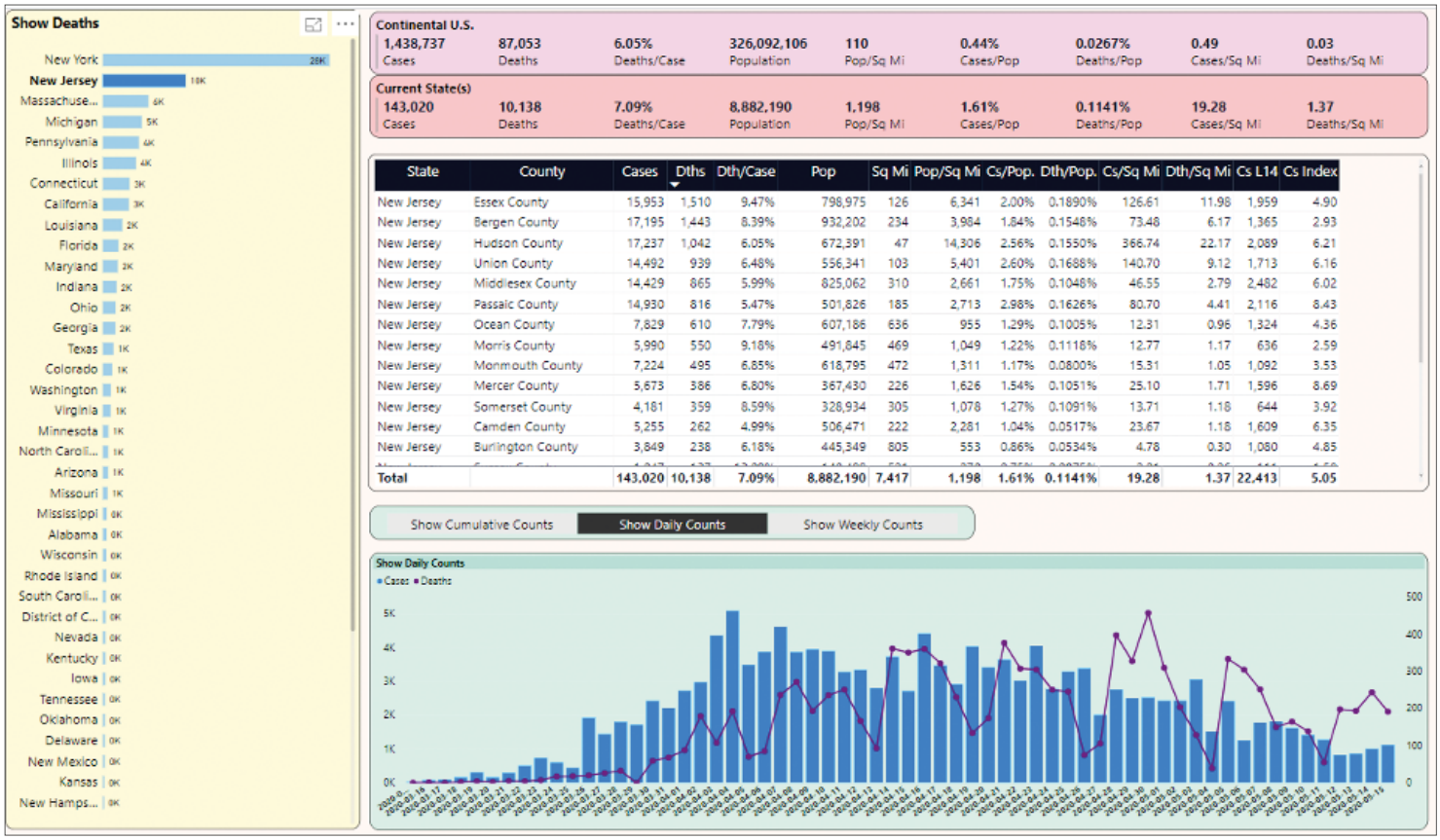


Figure 9: Comparing NJ state measures to U.S. measures and ranking NJ counties

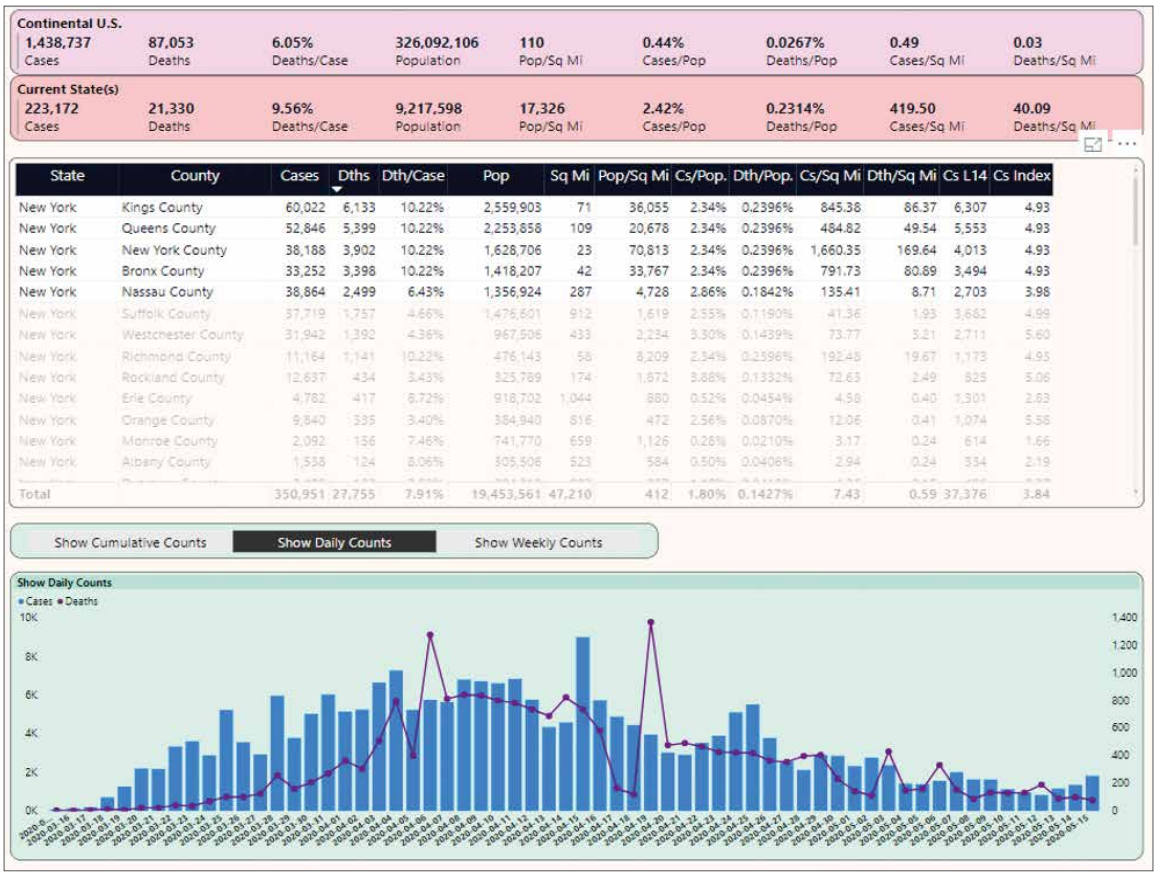
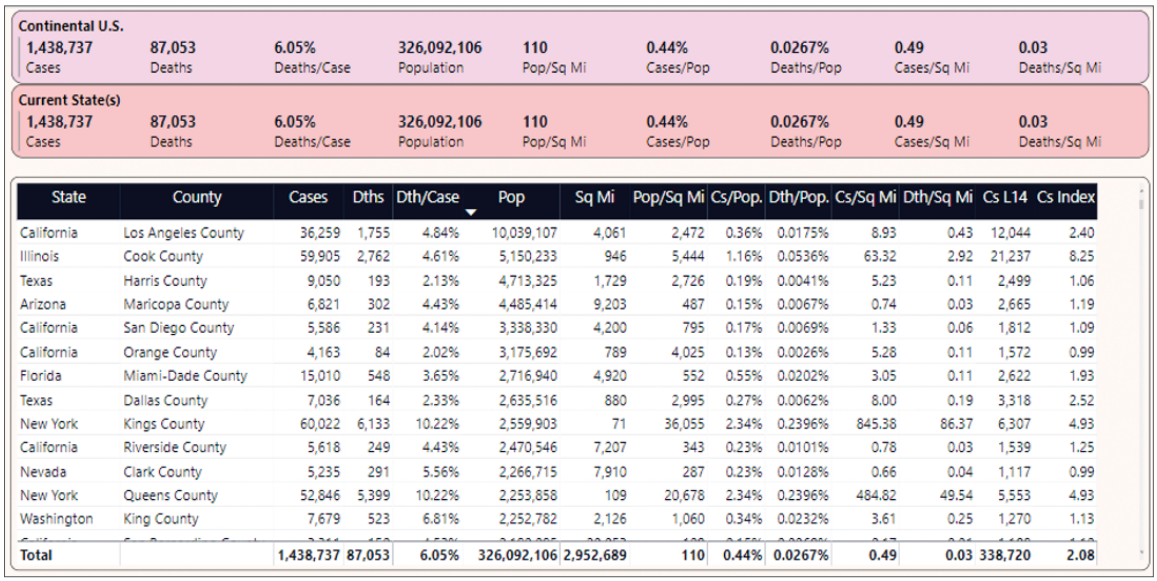


Figure 10: Sort NY counties by Deaths/Square Mile and then select first five



Continental U.S.

1,438,737 Cases	87,053 Deaths	6.05% Deaths/Case	326,092,106 Population	110 Pop/Sq Mi	0.44% Cases/Pop	0.0267% Deaths/Pop	0.49 Cases/Sq Mi	0.03 Deaths/Sq Mi

Current State(s)

1,438,737 Cases	87,053 Deaths	6.05% Deaths/Case	326,092,106 Population	110 Pop/Sq Mi	0.44% Cases/Pop	0.0267% Deaths/Pop	0.49 Cases/Sq Mi	0.03 Deaths/Sq Mi

State	County	Cases	Dths	Dth/Case	Pop	Sq Mi	Pop/Sq Mi	Cs/Pop.	Dth/Pop.	Cs/Sq Mi	Dth/Sq Mi	Cs L14	Cs Index
California	Los Angeles County	36,259	1,755	4.84%	10,039,107	4,061	2,472	0.36%	0.0175%	8.93	0.43	12,044	2.40
Illinois	Cook County	59,905	2,762	4.61%	5,150,233	946	5,444	1.16%	0.0536%	63.32	2.92	21,237	8.25
Texas	Harris County	9,050	193	2.13%	4,713,325	1,729	2,726	0.19%	0.0041%	5.23	0.11	2,499	1.06
Arizona	Maricopa County	6,821	302	4.43%	4,485,414	9,203	487	0.15%	0.0067%	0.74	0.03	2,665	1.19
California	San Diego County	5,586	231	4.14%	3,338,330	4,200	795	0.17%	0.0069%	1.33	0.06	1,812	1.09
California	Orange County	4,163	84	2.02%	3,175,692	789	4,025	0.13%	0.0026%	5.28	0.11	1,572	0.99
Florida	Miami-Dade County	15,010	548	3.65%	2,716,940	4,920	552	0.55%	0.0202%	3.05	0.11	2,622	1.93
Texas	Dallas County	7,036	164	2.33%	2,635,516	880	2,995	0.27%	0.0062%	8.00	0.19	3,318	2.52
New York	Kings County	60,022	6,133	10.22%	2,559,903	71	36,055	2.34%	0.2396%	845.38	86.37	6,307	4.93
California	Riverside County	5,618	249	4.43%	2,470,546	7,207	343	0.23%	0.0101%	0.78	0.03	1,539	1.25
Nevada	Clark County	5,235	291	5.56%	2,266,715	7,910	287	0.23%	0.0128%	0.66	0.04	1,117	0.99
New York	Queens County	52,846	5,399	10.22%	2,253,858	109	20,678	2.34%	0.2396%	484.82	49.54	5,553	4.93
Washington	King County	7,679	523	6.81%	2,252,782	2,126	1,060	0.34%	0.0232%	3.61	0.25	1,270	1.13
Total		1,438,737	87,053	6.05%	326,092,106	2,952,689	110	0.44%	0.0267%	0.49	0.03	338,720	2.08

Figure 11: Counties with the largest populations

even higher rates than the NY state figures as a whole. These are very densely populated counties with rates per square mile that reflect the severe crisis situation they have faced.

Now let's select New Jersey from the list of states on the left and continue to sort the counties on the right by Deaths/Square Mile. You can see in **Figure 9** that although Hudson County is third in cases with 819, it's first in NJ with 17.43 deaths per square mile. Also note that NJ's 1.02 deaths per square mile is roughly two-and-a-half times greater than NY's 0.39 deaths per square mile (from back in **Figure 7**).

Let's go back to **Figure 7** (New York) and on the right, click-CTRL-CLICK on the top five counties (**Figure 10**). Notice that the national average stays the same, but the second recap

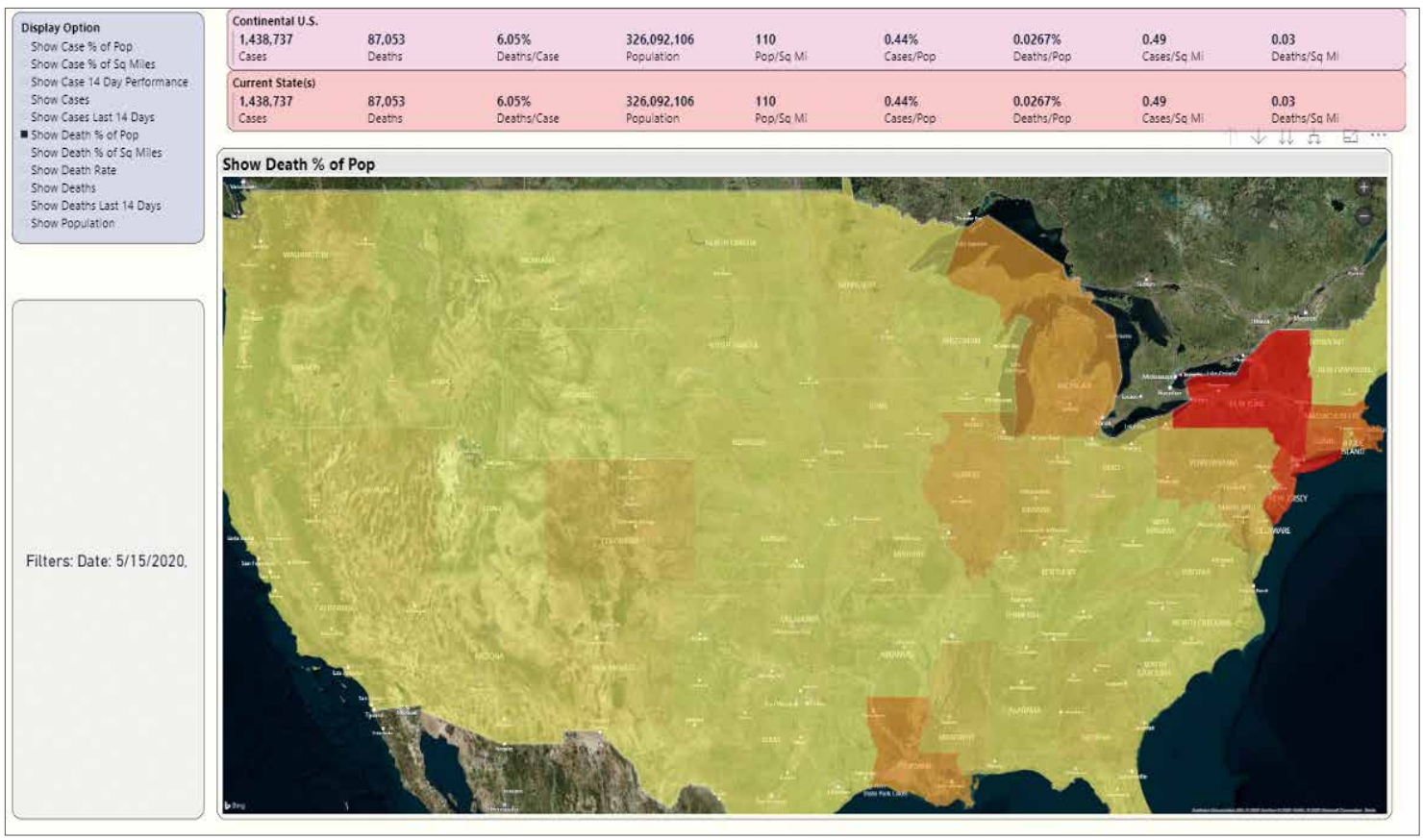


Figure 12: Map of 48 states filled with a rule based on the Dynamic Measure selected

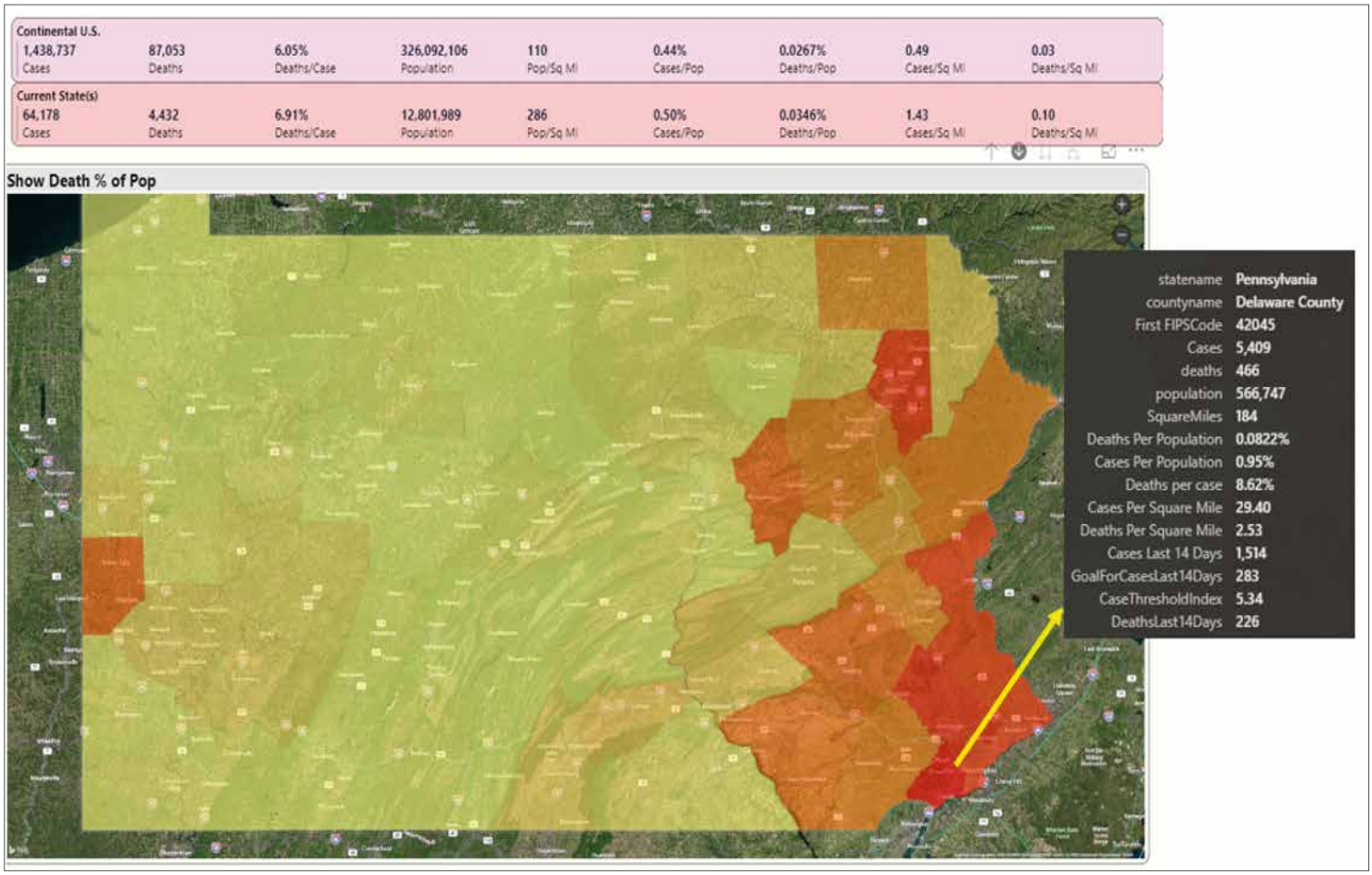


Figure 13: Breakout of Pennsylvania by country

line is filtered based on the five counties. This is exponentially sobering: The five counties that make up New York City collectively cover just under 400 square miles, yet account for about 25% of the nation's deaths.

Also notice that at the bottom of **Figure 10,** there's a daily chart of cases and deaths.

OK, let's stop and talk about a few things in the chart back in **Figure 7**, specifically the bar chart that ranked states by a dynamic measure I selected in a drop-down. In Power BI, you can use a DAX formula to determine which measure to use, based on the Measure option the user selects.

```
DynamicMeasure =
switch( values (  ShowMeasure[OptionName]  ),
 "Show Cases",  sum(CovidHistory[Cases]),
 "Show Deaths", sum(CovidHistory[deaths]) ,
 "Show Case % of Pop", [Cases Per Population] ,
 "Show Death % of Pop", [Deaths Per Population],
 "Show Death Rate", [Deaths per case],
 "Show Case % of Sq Miles",
  [Cases Per Square Mile], etc. )
```

Also, note the "filter recap" label, which is also a DAX formula that reads the current filter selections:

```
SelectedFilter = "Filters: " &
 if(isfiltered(   DateList[casedate]) ,
  "Date: " &
   CONCATENATEX(filters(DateList[casedate]),
      DateList[casedate]  ,
           ", ") & ", ","")  &

if(isfiltered(statemaster[statename] ),
  "State(s): " &
  CONCATENATEX(filters(
              statemaster [statename] ),
              statemaster[statename] ,
          ", ") & ", ","")
```

Finally, for general population and death statistics, **Figure 11** sorts the counties in the U.S. by population, so that you can see case/death rates for these counties.

Some observations:

- Cook County in Illinois has twice the population of either Kings County or Queens County in NY, yet the two NY counties are far more densely populated (36K and 20K per square mile versus 5.4K per square mile) and the difference in deaths per square mile is exponentially sobering.
- Additionally, the cumulative deaths in either Kings or Queens County NY is "roughly" near the sum of the other displayed counties combined.
- There are large counties in Texas, Arizona, and California with death rates/population below the national average.

Figure 14: Scatter chart for counties in PA. Notice the timeline axis.

Mapping

Figure **12** shows a map, based on cumulative numbers on 5/1. I've chosen "Death as a % of Population:". The country map fills states based on the measurement I selected in the drop-down.

Note that New York is the "reddest," with the highest death rate per population in the U.S.

However, note that Pennsylvania, Louisiana, Georgia, and Michigan have a slightly darker shade than other states. I'll take a look at those states one at a time.

Because I live in Pennsylvania, I'll click on that state, and Power BI will plot the counties by the metric I've chosen ("Death as a % of Population), in **Figure 13**.

This has largely been the story in Pennsylvania for over a month. The southeastern part of the state (closest to New Jersey and NY) have the highest percentages. Inside each of the counties are incredibly sad and tragic stories. Lehigh County (Allentown) has reported that over 70% of deaths have occurred in long-term care facilities.

Note that I've placed the mouse over Delaware County, which is just west of Philadelphia county. I've got a tooltip with all the available information. Later in the article, in the Playbook section, I talk about the value of tooltips. Delaware County has percentages that are generally worse than the overall national averages.

In a moment, I'll look at a few other states (Louisiana, Georgia, and Michigan). They have counties and scenarios that

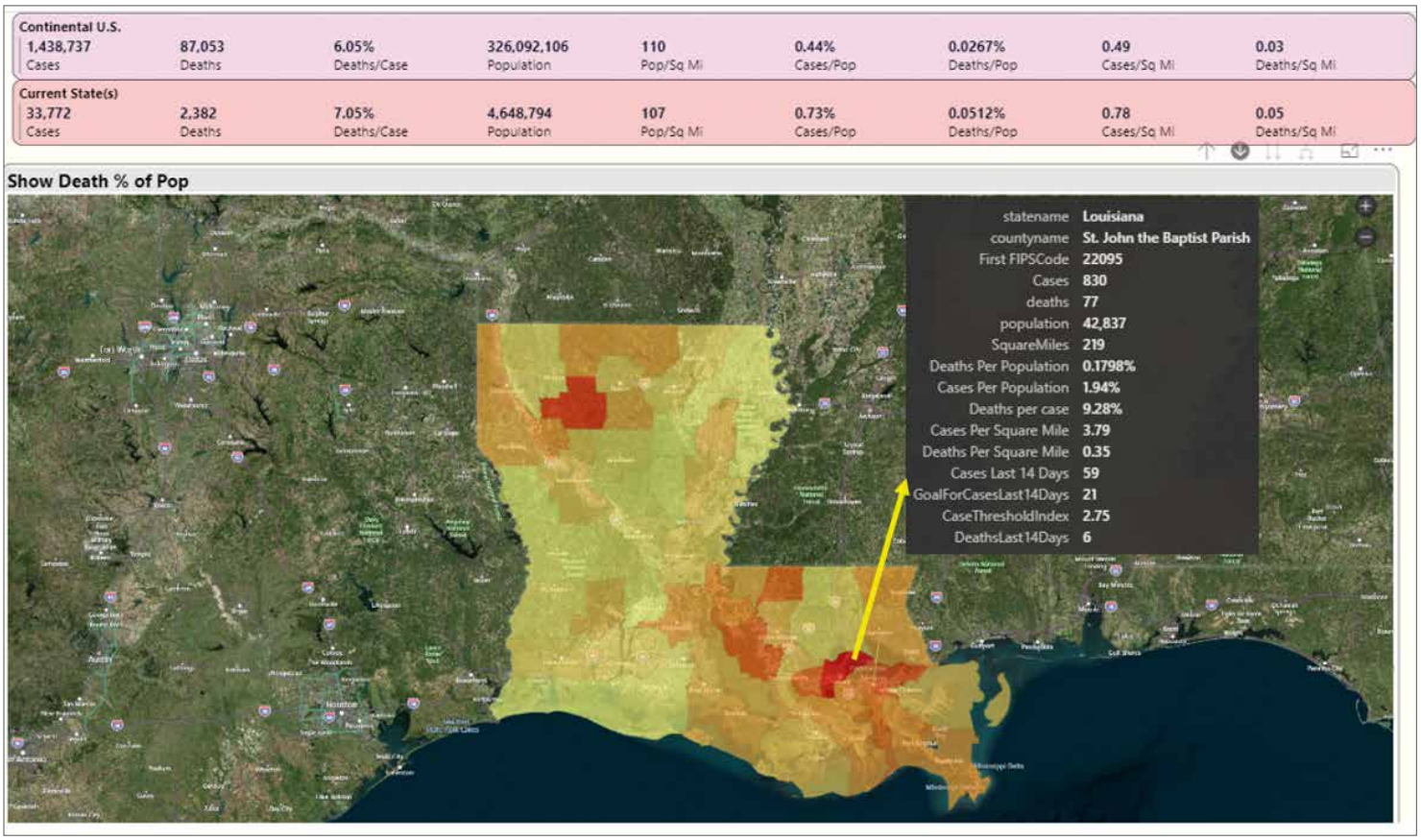


Figure 15: State of Louisiana, tooltip for St John the Baptist Parish

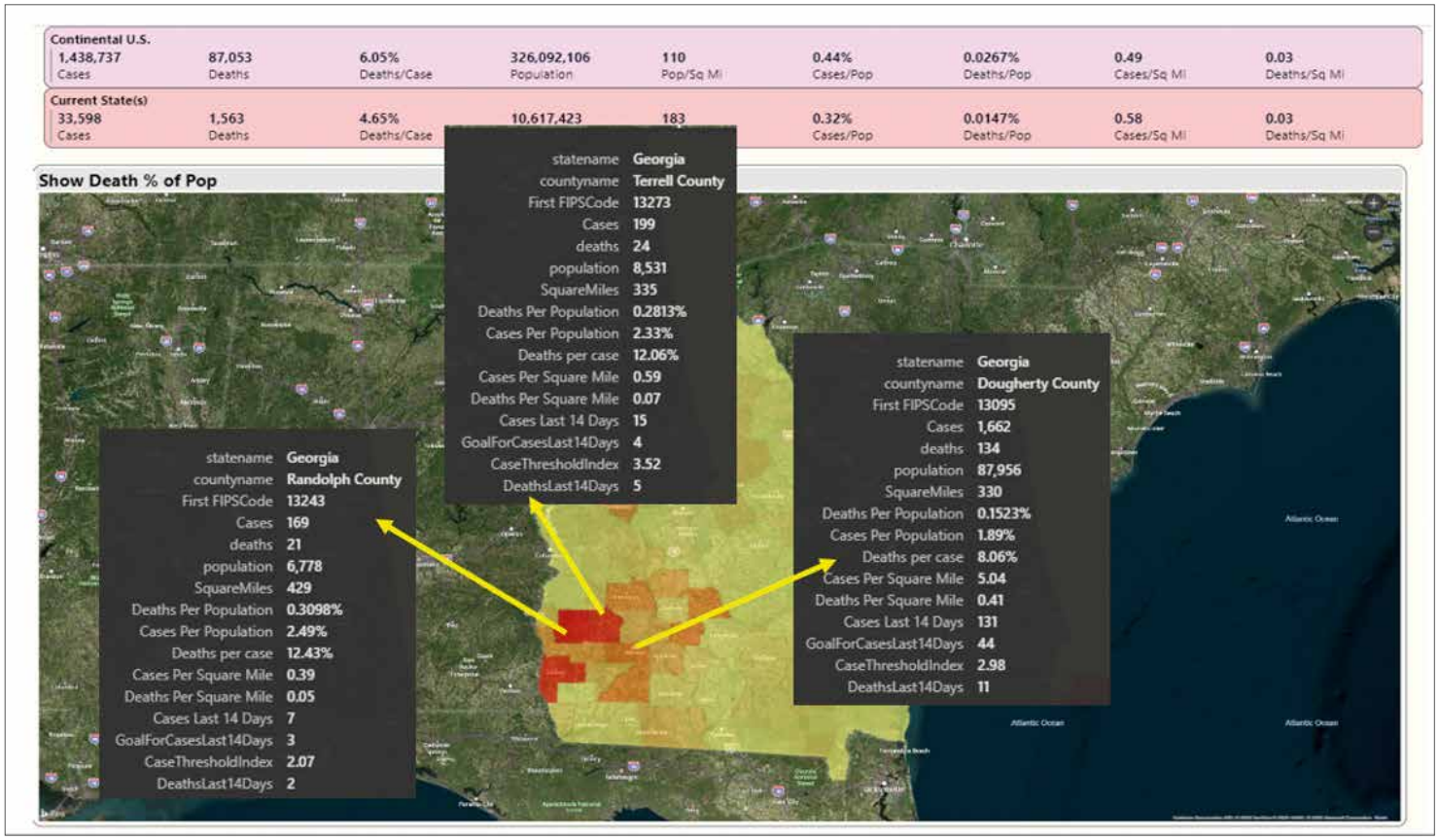


Figure 16: Georgia's Dougherty County and neighboring Randolph and Terrell Counties

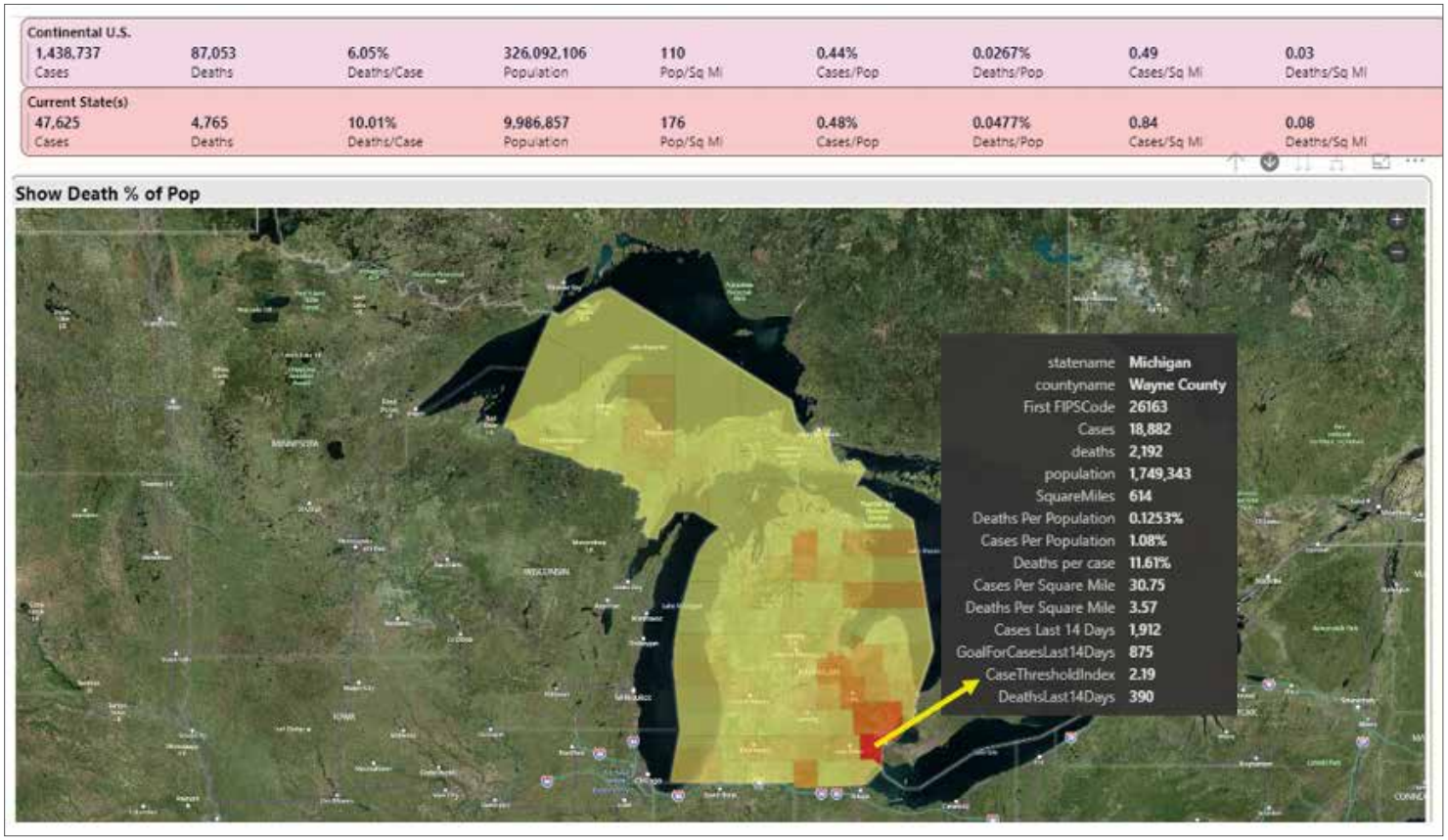


Figure 17: Michigan, with focus on Wayne County (Detroit)

you might or might not have heard about on the national news, but are still far above the national averages.

But before then, you might want to know something. How have deaths progressed in Pennsylvania? I can provide a right-click drill-through option to show a scatter graph that plots population and deaths by county, along with a daily timeline (**Figure 14**):

At the top of **Figure 14**, note that on April 7th, the Y-axis for death count was nearly the same for most counties. But then as **I slide the date axis at the bottom forward**

State	County	Cases	Dths	Dth/Case	Pop	Sq Mi	Pop/Sq Mi	Cs/Pop.	Dth/Pop.	Cs/Sq Mi	Dth/Sq Mi	Cs L14	Cs Index
Georgia	Randolph County	169	21	12.43%	6,778	429	16	2.49%	0.3098%	0.39	0.05	7	2.07
Georgia	Terrell County	199	24	12.06%	8,531	335	25	2.33%	0.2813%	0.59	0.07	15	3.52
Georgia	Early County	233	28	12.02%	10,190	511	20	2.29%	0.2748%	0.46	0.05	19	3.73
New York	Kings County	60,022	6,133	10.22%	2,559,903	71	36,055	2.34%	0.2396%	845.38	86.37	6,307	4.93
New York	Queens County	52,846	5,399	10.22%	2,253,858	109	20,678	2.34%	0.2396%	484.82	49.54	5,553	4.93
New York	Bronx County	33,252	3,398	10.22%	1,418,207	42	33,767	2.34%	0.2396%	791.73	80.89	3,494	4.93
New York	New York County	38,188	3,902	10.22%	1,628,706	23	70,813	2.34%	0.2396%	1,660.35	169.64	4,013	4.93
New York	Richmond County	11,164	1,141	10.22%	476,143	58	8,209	2.34%	0.2396%	192.48	19.67	1,173	4.93
New Jersey	Essex County	15,953	1,510	9.47%	798,975	126	6,341	2.00%	0.1890%	126.61	11.98	1,959	4.90
New York	Nassau County	38,864	2,499	6.43%	1,356,924	287	4,728	2.86%	0.1842%	135.41	8.71	2,703	3.98
Louisiana	St. John the Baptist Parish	830	77	9.28%	42,837	219	196	1.94%	0.1798%	3.79	0.35	59	2.75
New Jersey	Union County	14,492	939	6.48%	556,341	103	5,401	2.60%	0.1688%	140.70	9.12	1,713	6.16
New Jersey	Passaic County	14,930	816	5.47%	501,826	185	2,713	2.98%	0.1626%	80.70	4.41	2,116	8.43
New Jersey	Hudson County	17,237	1,042	6.05%	672,391	47	14,306	2.56%	0.1550%	366.74	22.17	2,089	6.21
New Jersey	Bergen County	17,195	1,443	8.39%	932,202	234	3,984	1.84%	0.1548%	73.48	6.17	1,365	2.93
Georgia	Dougherty County	1,662	134	8.06%	87,956	330	267	1.89%	0.1523%	5.04	0.41	131	2.98
Total		**1,438,737**	**87,053**	**6.05%**	**326,092,106**	**2,952,689**	**110**	**0.44%**	**0.0267%**	**0.49**	**0.03**	**338,720**	**2.08**

Figure 18: Sort the grid on the first dashboard page by Death Rate per Population

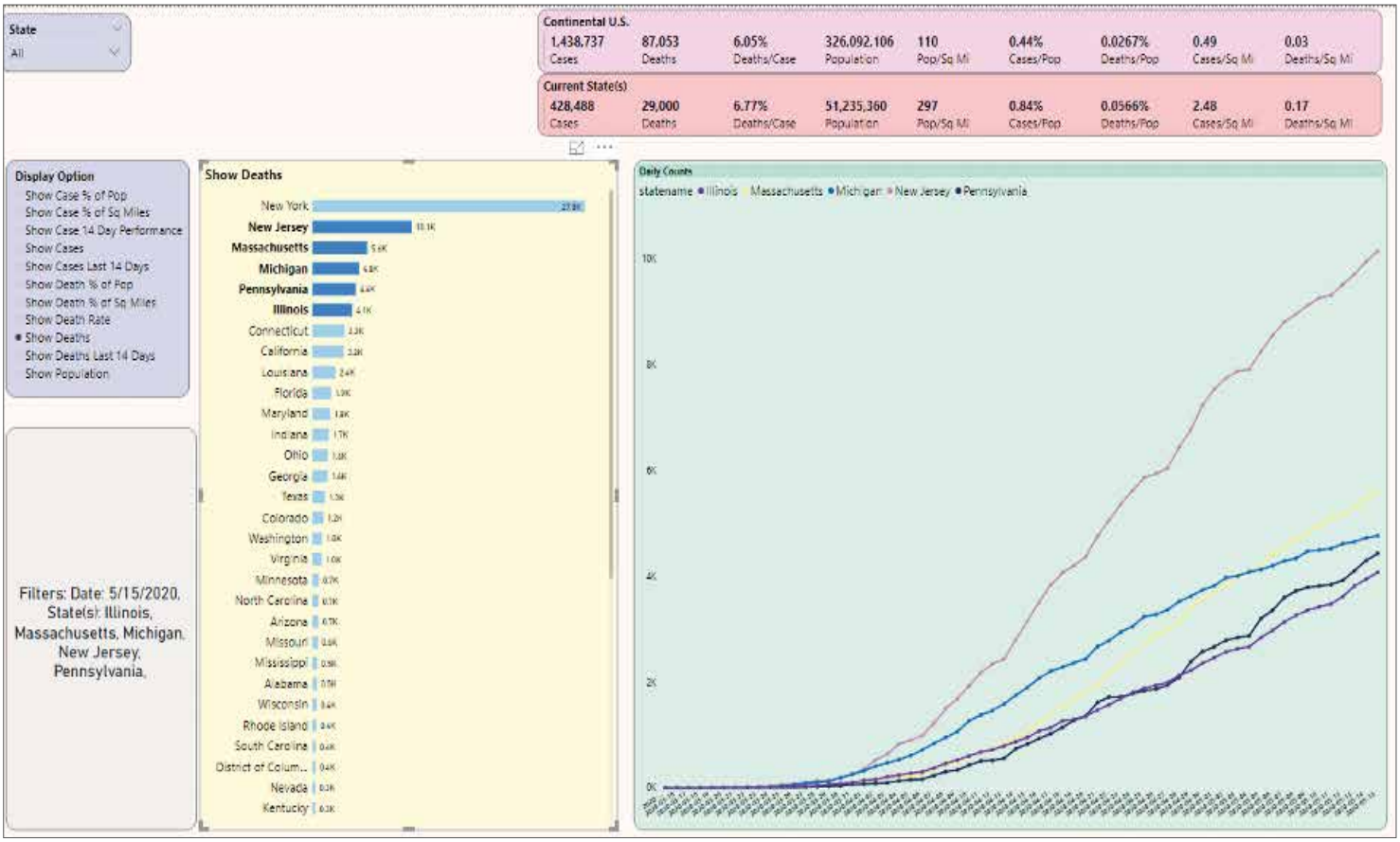


Figure 19: State-by-State comparison of a metric (Deaths) for specific states

by day, I interactively see that the death count in Philadelphia has risen sharply. And even though Allegheny county is the second most populated county in PA, the deaths in smaller counties like Bucks, Delaware, and Lancaster are higher.

Let's go back to the national map and then drill into Louisiana. Note: Louisiana is broken out by Parish, which is roughly the equivalent of a county. Louisiana was originally part of Spain and then France, and those countries divided their lands according to church parish boundaries. Even after Louisiana became part of the U.S., they continue to refer to regions as parishes and not counties.

In **Figure 15**, you can see that St John the Baptist Parish has the highest death rate (red) in the state. Even though it isn't a large region (about 43K residents), the case and death rates are very high: as you'll see in a minute, some of the highest in the nation.

In **Figure 16**, I'll take a look at Georgia. I lived in Atlanta in the 1990s and travelled to nearly every county health department as part of a state-wide automation project. I was surprised and saddened to learn that Dougherty County, a few hours south of Atlanta with a population of about 88K, had been severely hit. The neighboring county of Randolph, with just 6.7K residents, has also been severely hit, with one of the highest cases and death/population percentages in the nation. Several news sources reported that many of the deaths occurred in a nursing home that received essential services from neighboring Dougherty County. Additionally, Terrell County, just east of Randolph County, has death per population figures that are much higher than the national average.

In **Figure 17**, I'll look at Michigan. I've spent time in Michigan the last 15 years, both in Traverse City and also in Dearborn. Right now, Wayne County (both Detroit and Dearborn) has also faced very high case and death rates. You can see from **Figure 17** that Wayne County has nearly half the deaths in Michigan (2,192 vs. 4,765) despite occupying only 614 square miles.

Now that you've seen some county scenarios for a handful of states, let's see how many of them show up when I sort all counties in the U.S. by death/population. Let's go back to the grid on the first page of the dashboard, make sure all states on the left are selected, and sort the counties on the right by Death/Population (**Figure 18**). Although you can see several counties in NY in the list, you also see several counties in Georgia at the top of the list, as well as St John the Baptist Parish in Louisiana. And although not pictured, Wayne county in Michigan is also high on the list if you scroll down a few more counties.

State Comparisons

You know that New York has, by far and away, the highest number of cases and deaths. But for the next five states based on deaths, how much do they account for the nation's total and how do they compare to each other?

In **Figure 19**, I've gone to the tab for state comparisons and I've selected states two through six (New Jersey, Michigan,

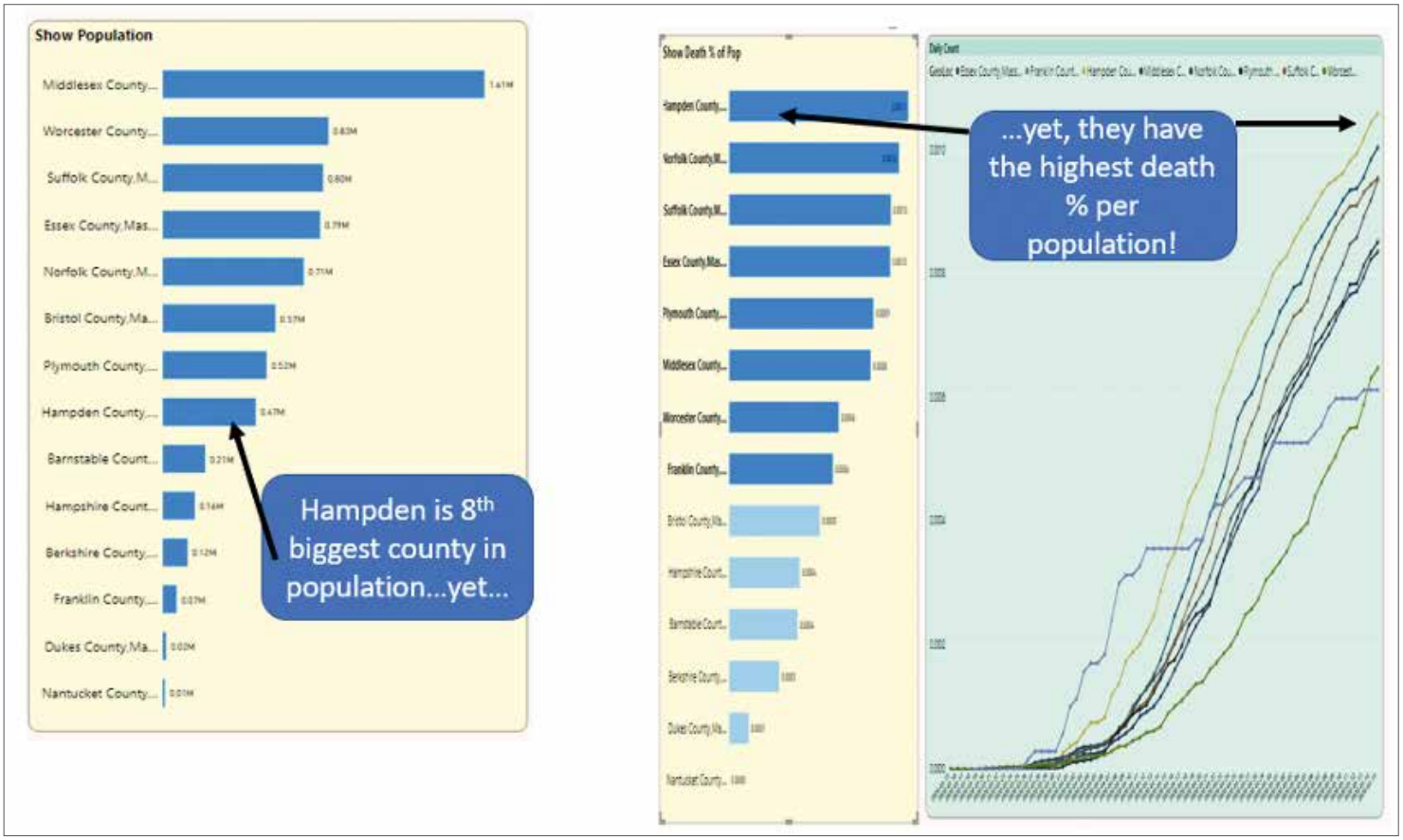


Figure 20: Massachusetts counties, first by population and then by Death Rate

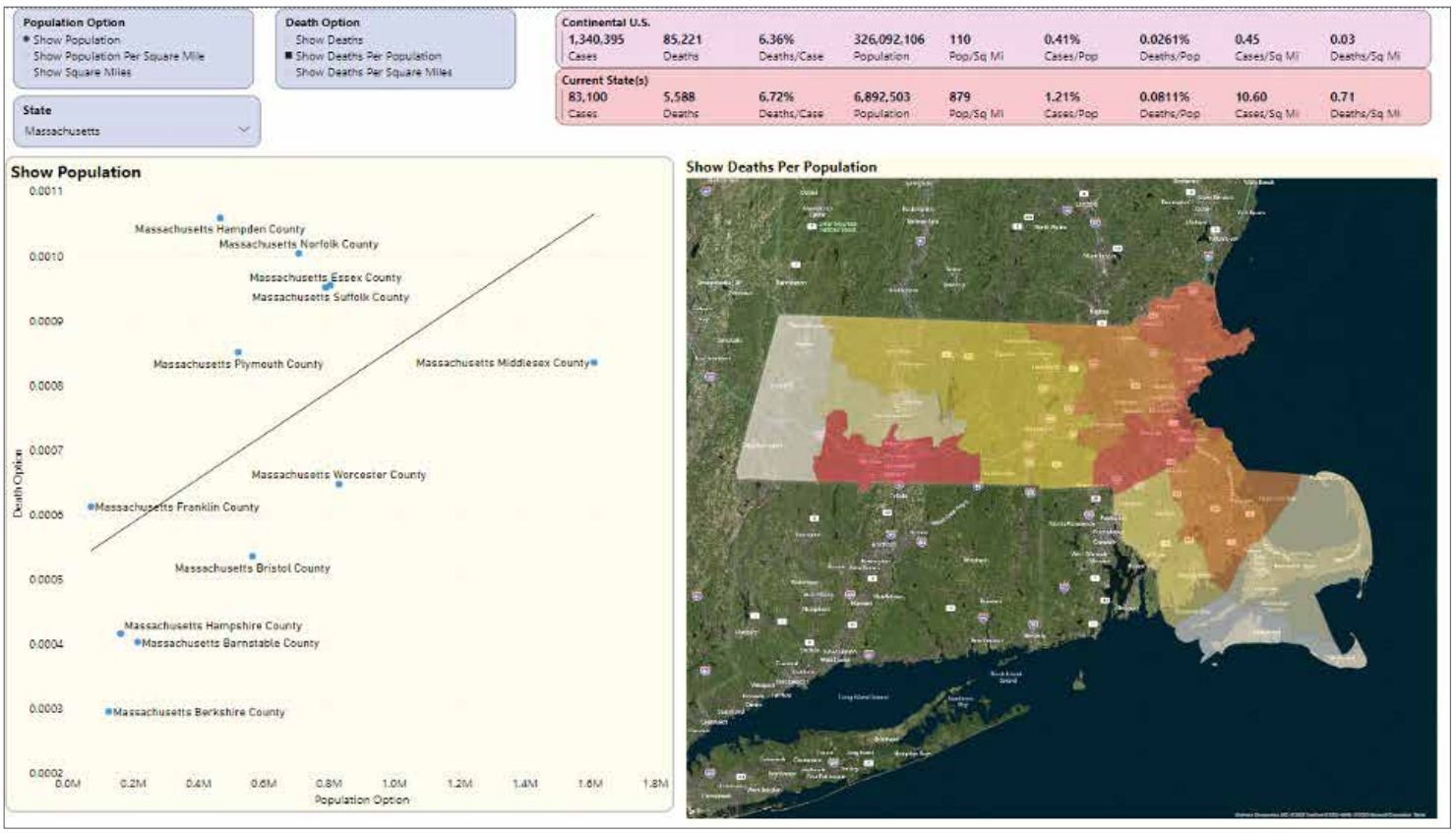


Figure 21: Alternate view of Hampden county: a scatter chart plotting population vs. death rate

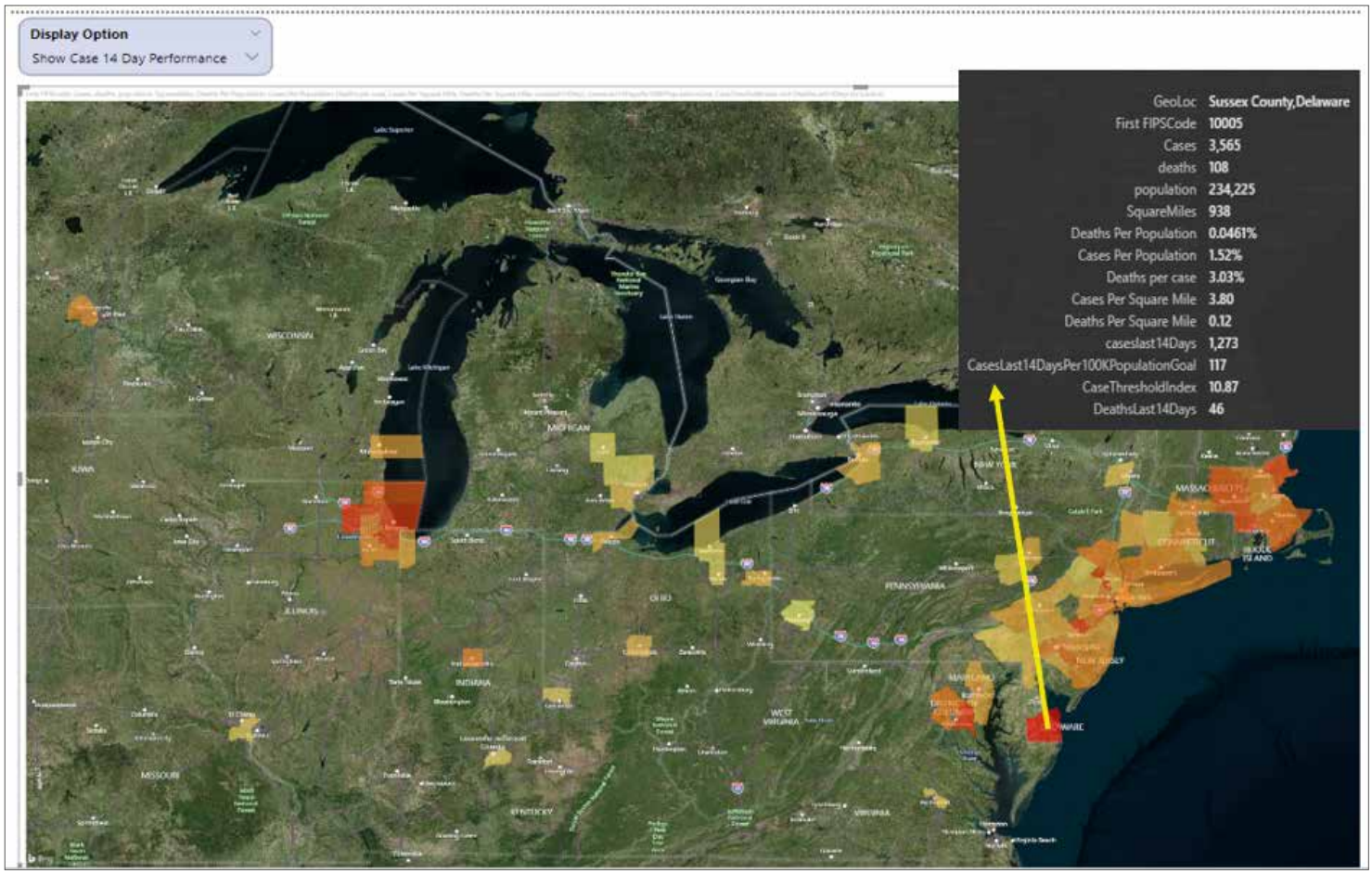


Figure 22: Counties with the worst performance index for re-opening based on a 14-day case count

Massachusetts, Pennsylvania, and Illinois.) You can see that in the last few weeks, deaths have spiked in New Jersey at a higher rate than other states. Also, of interest, Massachusetts is close to Michigan in cumulative deaths, even though Michigan has roughly three million more residents. Deaths in Massachusetts have risen sharply in the last month.

Here's a question: Is the rise in Massachusetts directly correlated to population? As it turns out, not always. In **Figure 20**, on the far left, Hampden County in Massachusetts is the eighth largest county in Massachusetts in terms of population but has the highest death per population statistic.

It occurred to me that another visualization could be a scatter chart where the user could select the option for one axis (Population, or Population Per Square mile, etc.) and then the other axis (Deaths, Deaths per population, etc.).

The scatter chart in **Figure 21** is just another way to show that Hampden County is far from the most populated county but has the highest death rate per population. In the scatter chart, Hampden represents the furthest "negative" distance from the regression line on the relationship between population and deaths per population. **Scatter charts help us to see distribution of values and relationships between two variables.**

Assessment of Readiness

Finally, if there were a nationwide goal of fewer than 50 new total cases per county over the last 14 days for every 100K residents in that county, which counties would be the best (or worst) position for that? (Note: I realize that many will have opinions on the parameters for such a goal. My objective here is simply to show the mechanics of a KPI).

Figure 22 plots that measure and find that Sussex County in Delaware has had 1,273 cases over the last 14 days. They have a population of 234,225. If you applied that metric goal above, you'd want no more than 117 total cases over the last 14 days.

Okay, so that's 1,273 divided by 117, which is an index of roughly 10.87. Basically, according to the rules of this KPI goal, cases over the last 14 days are roughly 10.87 times greater than they should be, in order to start re-opening. That represents one of the higher indexes in the country in terms of readiness. In other words, the lower the index, the better.

If you go back to the grid on the opening page and add in the 14-day case total and the performance index, you can see that, of the largest counties in the U. S., some are close to (or under) the threshold and some are not **(Figure 23)**.

For instance, Harris County in Texas has had 2,499 cases in the last 14 days (relative to May 15). They have a population of 4.7 million people. Using the rule of no more than 50 cases per 100K residents, that means a possible indicator for reopening would be no more than ((4.7 million /

State	County	Cases	Dths	Dth/Case	Pop	Sq Mi	Pop/Sq Mi	Cs/Pop.	Dth/Pop.	Cs/Sq Mi	Dth/Sq Mi	Cs L14	Cs Index
California	Los Angeles County	36,259	1,755	4.84%	10,039,107	4,061	2,472	0.36%	0.0175%	8.93	0.43	12,044	2.40
Illinois	Cook County	59,905	2,762	4.61%	5,150,233	946	5,444	1.16%	0.0536%	63.32	2.92	21,237	8.25
Texas	Harris County	9,050	193	2.13%	4,713,325	1,729	2,726	0.19%	0.0041%	5.23	0.11	2,499	1.06
Arizona	Maricopa County	6,821	302	4.43%	4,485,414	9,203	487	0.15%	0.0067%	0.74	0.03	2,665	1.19
California	San Diego County	5,586	231	4.14%	3,338,330	4,200	795	0.17%	0.0069%	1.33	0.06	1,812	1.09
California	Orange County	4,163	84	2.02%	3,175,692	789	4,025	0.13%	0.0026%	5.28	0.11	1,572	0.99
Florida	Miami-Dade County	15,010	548	3.65%	2,716,940	4,920	552	0.55%	0.0202%	3.05	0.11	2,622	1.93
Texas	Dallas County	7,036	164	2.33%	2,635,516	880	2,995	0.27%	0.0062%	8.00	0.19	3,318	2.52
New York	Kings County	60,022	6,133	10.22%	2,559,903	71	36,055	2.34%	0.2396%	845.38	86.37	6,307	4.93
California	Riverside County	5,618	249	4.43%	2,470,546	7,207	343	0.23%	0.0101%	0.78	0.03	1,539	1.25
Nevada	Clark County	5,235	291	5.56%	2,266,715	7,910	287	0.23%	0.0128%	0.66	0.04	1,117	0.99
New York	Queens County	52,846	5,399	10.22%	2,253,858	109	20,678	2.34%	0.2396%	484.82	49.54	5,553	4.93
Washington	King County	7,679	523	6.81%	2,252,782	2,126	1,060	0.34%	0.0232%	3.61	0.25	1,270	1.13
Total		**1,438,737**	**87,053**	**6.05%**	**326,092,106**	**2,952,689**	**110**	**0.44%**	**0.0267%**	**0.49**	**0.03**	**338,720**	**2.08**

Figure 23: All counties, sorted by 14-day Case per Population Performance Index

100,000) multiplied by 50), or roughly 2,350 new cases. As a result, they are slightly above 1 for the index.

Playbook: Presentation Layer

Now that I've gone through the application, I'll cover some of the items I've borrowed from my "playbook," both on the Presentation and Non-Presentation layer. I'll start with items I often consider on the presentation end, whether talking about COVID statistics or inventory/order book dashboards or financial dashboards:

Shared Filters

Recently, I needed to filter on data from a website and found that I couldn't. The data element I needed was a pretty common one for the business context and had a fairly short list of discrete values. Unfortunately, the website didn't permit me to filter. To make matters worse, the site made it difficult for me to pull back raw data and filter it myself.

I tend to be a pretty grumpy user when I can't get what I think shouldn't be realistically difficult, and that's precisely why I try to be very sensitive to user filtering needs. I also get cranky when I set a filter on one page and there's no option to persist that filter on subsequent pages where the specific filter would hold the same context.

There weren't many filters in this COVID dashboard report, but I did try to make as much available as possible and persisted common filters (like current snapshot date) across pages.

Keep the Door Open for Exporting Raw Detailed Data

Granted, applications can only drill-down so far. Having said that, don't assume that users will be satisfied with summary information. Users will want details, and they'll often want to export those details to Excel to do their own analysis.

Fortunately, Power BI provides native ability for an end user to export the results of a visualization (i.e., a chart, a table, etc.) to a CSV file. Every Power BI visualization has a runtime option in the upper-right hover menu to export data.

Content Reward

Just like the old TV show where a contestant won by "naming that tune in four notes," a successful reporting application provides meaningful content reward with a minimal number of mouse-clicks. In my career, I've built what I thought were great reporting applications, only to learn that users found them too difficult to use. Users are busy and don't have time to learn all sorts of system intricacies. This is profoundly seen in Apple products. The more information you provide to users with a minimal amount of effort, the more they'll want to use your applications.

I've tried to demonstrate this theme in **Figures 7** through **9**, where someone could see country and specific state/county metrics in one glance. I've also tried to demonstrate this in tooltips to show as much supporting data as possible.

Dynamic Measures

If you have a reporting application with many key metrics, it's tempting to create a page or page segment for each measure. In some cases, you can alternatively display all the possible measures in a drop-down and then plot the selected measure dynamically in a chart.

> Just like the old TV show where a contestant won by "naming that tune in four notes," a successful reporting application provides meaningful content reward with a minimal number of mouse-clicks.

In this application, **Figure 7** showed where you can rank states by a variable metric and **Figure 8** shows where you can show daily counts or cumulative counts by day. I defi-

nitely recommend considering this approach to avoid repeating the same visualization across many pages.

Again, Power BI doesn't currently seem to have any elegant way of specifying dynamic attributes (such as allowing a user to show geographies or products or GL codes, etc.). Some have suggested a workaround of exploding/normalizing the result set and joining to a bridge table of attribute definitions. If the result set isn't large, this **can** work, but isn't practical for larger applications. Hopefully Microsoft will provide the ability to set dynamic attributes via a simple DAX formula, in the same way we can do it now with measures.

Maps

Some developers love maps and want to show them to users (whether users ask for them or not). Other developers will avoid them at all costs. You can use them to create great visualizations, but anyone who's worked with mapping components knows that they can be challenging. I've made moderate but not heavy use of mapping here. I think they provide a nice touch and serve a particularly great role in presentations. However, you need to learn the rules and any nuances for what geography elements (names or geospatial/latitude-longitude points) the mapping visualization expects. You also need to be careful about overloading a map, which can take much longer than a few seconds.

This application has only used maps as a visual launching point to talk about certain states and to help the user easily identify counties. Other applications use maps for more intricate geographic definitions than what I've provided here, such as distances between grocery stores. (Ironically, the recent discussion of contact tracing opens discussions about far more elaborating mapping than what I've provided here).

Tooltips

A long time ago, I had a boss who was adamant about providing as much tooltip content as possible. He felt that any calculated field should also immediately show all the values involved in the calculation. Periodically he'd go through my applications, print out any screen shots where the tooltip didn't sufficiently provide all the background information for a calculation, and leave the printouts on my desk (and also email everyone in the company). He was brutal. I didn't care for the tactics, but he was almost always correct.

Figures 13, 15-17, and **22** all show examples of tooltips. Perhaps I went overboard, but it's usually better situation to remove unnecessary content than to add it.

Final note: Microsoft also has a feature in Power BI called Report Tooltips. You can define a separate report page and make it available as a tooltip for a plotted page. I've seen applications that have gone overboard with this concept, but it provides a powerful way to give context to a plotted point on a chart, far more than a regular tooltip list of values. In my next article, I'll cover this in detail.

Cross-Filtering

Power BI provides some fantastic capabilities for cross-filtering. If you're not familiar with the term, imagine if you have a chart on the left of Sales by Product and the chart on the right of Sales by Week. If you click on a product on the left, the chart on the right auto-refreshes based on the product(s) you selected. Conversely, if you click on specific weeks on the right, the chart on the left is auto-refreshed based on the weeks you selected.

This is a great feature that most users expect. The only catch is that you want to test all the possible behaviors and selection options. If you've applied visual-level filters for things like Top 25 Products, the cross-filtering on the right might not behave exactly the way you expect.

A few years ago, I wrote here in CODE Magazine about how to mimic this feature in SQL Server Reporting Services, using invisible hot parameters. You can find the solution in the May/June 2016 issue of CODE Magazine, tip #3 (https://www.codemag.com/Article/1605111). Although I still use that feature today in SSRS, there's no question that Power BI's native feature for cross-filtering is much snappier.

It's difficult to see the impact of cross-filtering in a printed article, but certainly the ability to click a state back in **Figures 7** and **8** and immediately see the relevant counties on the right is an example of cross-filtering.

Easy Way to See "Best/Worst 25"

Going back the theme of "I can name that tune in X mouse-clicks," it's very important for a user to be able to see a list of top/bottom values with little effort. The "best" or "worst" of a metric is often key to any analysis: Users will appreciate seeing this information easily.

My reporting applications have all sorts of "Top/Bottom 25" lists of worst orders by margin, best sales reps by margin, etc. Often business users will be quite vocal up front about wanting to see this, and never hesitate to raise it as a possible feature.

In this COVID reporting application, the list of counties in the figures for general statistics showed how you could easily get the top N or bottom N based on a metric. Power BI makes this particularly easy by allowing native sorting on a column in ascending or descending order.

Recap of Main Numbers: Your Version of "The Magic Wall"

I have saved the most esoteric top for last. If you watch news shows on television, some stations have analysts who show interactive maps of voting trends, voting demographics, etc. I frequently watch these shows, but not for the reason you might think. More than anything, I'm utterly fascinated with the use of technology to tell a story. In particular, CNN's John King is a master in using the multi-touch collaboration wall (nicknamed "The Magic Wall") to tell a story.

When you create dashboard pages to show company financial performance, company costs, or sales trends by product/market, essentially you have a story to tell regarding the data. Maybe margin rates are down this month because you had to offer lower prices and could not lower the costs as well. Maybe margin rates are down because you wound up selling more of lower margin products than higher margin products. Bottom line, some aspect of overall performance is up or down because of this and this and this. The factors could be simple, or more complicated.

A business intelligence application is the means by which you tell that story. This means three things:

Some Business Buzzwords are Worth Remembering

Some business buzzwords make me want to reach for a bottle of Pepto; however, others are important. Here's one from the latter category: SO THAT. We do this SO THAT we can accomplish something. In this case, we are trying to take large amounts of data (whether organized neatly or not) and present a story of what happened to the business.

- The data in the story needs to be accurate.
- The presentation of the data needs to be clear and representative of how you label/annotate it on the screen.
- Regardless of whether you use a bar chart, scatter chart, or just a sorted table, the visualization needs to be simple enough to be obvious, but not so simple that it prevents the user from being able to get more information.

On that last note, there can be instances where some think a vertical bar chart is better than a horizontal bar chart. Sometimes people use bar charts when line charts might arguably be better. (Pro tip: Don't use a pie chart if you'll have more than a few entries!)

Every once in a while, maybe before a major release, pretend like you're a news analyst using your tool to explain why some aspect of the business is up or down. Here's an even crazier suggestion: Capture a five-minute video and watch it. In the same way you do code reviews and design reviews, a presentation review can flush out quirks.

Playbook: Non-Presentation Layer

Now that I've talked about some of the presentation plays, let's talk about the playbook for the non-presentation layer. I mentioned at the beginning of this article that I've written past articles about ETL components in data warehousing. That's one aspect (and a BIG one) of all the work that the user doesn't directly see. However, there are others. Let's look.

Trends and Variances, Yes! But What Type?

Trend analysis and variance analysis can mean different things to different groups. Sometimes people just want to see the biggest increases or decreases from month to month. Other times people want to see when a product suddenly sells far more (or less) than the average over the last six months.

> Reporting on cases and deaths gives you one picture. But aligning those measures with population density frames a better picture. Similarly, a dashboard of shipment trends gives you one picture, but integrating the upcoming order booking and inventory turnover trends gives a much more insightful and compelling picture.

I have an application that shows the standard shipping rate for a city, and the different carrier rates, ranked by the highest carrier variance over the average. This shows which carriers are charging the most, relative to the average. I have another application that shows, month to month, the highest margin rate increase/decrease by geography/product. Out of dozens of products and hundreds of customers, for those with more than X dollars of sales to begin with, what single customer/product had the biggest increase or decrease in margin from month to month?

The KPI at the end is an example of a trend-basic metric, as I'm evaluating if counties are ready to re-open based on 14-day trends.

Spotlight: Combining Metrics to Form a Picture

This is one of the most important elements of valuable analytic reporting applications: bringing together different elements to help form a narrative. I cannot stress this enough:

Reporting on cases and deaths from COVID-19 gives you one picture. But aligning those measures with population density will frame a better picture. Similarly, a dashboard of shipment trends gives you one picture, but integrating the upcoming order booking and inventory turnover trends gives a much more insightful and compelling picture.

> Know the gaps and quirks in your source data. I had to spread out NY cases by county based on population and with unknown counties by population.

Where this can get tricky is when the elements don't share the same level of granularity. Metrics won't align properly if they don't share a common business context. Sometimes it's obvious and sometimes it's subtle. I've seen applications that mixed aggregations (both summary and averages) at different levels and gave a misleading picture. Ralph Kimball wrote profoundly about this in his data warehouse books. Even though he has retired, you can still find links to his books and design tips at https://www.kimballgroup.com/. I highly recommend that all data warehouse and business intelligence professionals read Kimball's works, even if you only implement a small portion of his ideas in your work. In particular, the book "The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling" truly deserves a rating of "5 out of 5 stars, a must-read".

Scheduled Refreshes

Another theme to stress in any reporting application is defining and annotating the data refresh schedule. Although there are a growing number of "real-time" analytic applications, many still work on a scheduled refresh that could be hourly, daily, twice a day, etc. In this instance, the COVID-19 reporting application is simple: The data is updated once a day from the GitHub source (typically by noon, Eastern Standard time) for the prior business day.

Materializing Data from a Mode; Table or View?

Opinions vary on this topic, but I try to keep views materialized and fairly simple. Nearly any database feature can be misused, and I've seen views misused more than any other database object. When effectively possible, I'd rather generate a flat report structure table for a set of visualizations.

> No one should overlook a tried-and-true data profiling approach: the good, old fashioned pair of eyes (and extra pair of eyes).

ETL, Data Lineage, and Data Munging

I've written on this topic in prior CODE articles (particularly back in 2017) and so I don't want to repeat large amounts of text. If you go to my CODE Magazine bio page (https://codemag. com/People/Bio/Kevin. Goff), there are multiple articles on Data Warehouse ETL techniques. Once again, I pay a lifetime of homage to the great works of Ralph Kimball, who wrote extensively on ETL and related activities.

In this reporting exercise, most of the ETL was basic. The only real issues I encountered with the data were that some state death counts had a county of "Unknown" and a few counties in the data feed did not have a FIPS code. I dealt with the former by spreading out the "unknown" case/death counts across the counties in that state by population. Admittedly, I took a shortcut with a hard-wired case statement to deal with the few counties that did not have a FIPS code. In an actual application, I'd want to use a lookup table.

Know the gaps and quirks in your source data. Here, I had to spread out NY cases by county based on population. I also had to spread out state cases with unknown counties by population. Recognize and account for it, document it, and make your key users aware of it.

There are great data profiling tools to spot data anomalies. No one should ever overlook their value—just like no one should overlook a tried-and-true data profiling approach: The good old-fashioned pair of eyes (and extra pair of eyes).

Validation with Other Systems

I validate numbers constantly. Maybe it's part of my obsessive nature. When I started producing output for this reporting application, I checked other websites to make sure my numbers were the same or at least reasonably close.

Within an organization, rarely will an application exist in isolation of others. People will compare your output to other systems, general ledger numbers, etc. I highly encourage people to stay on top of data validation, as difficult as it can be.

Here's an example: Over the last year, I've been working on a costing application. Users of this application compare the results against data in their Great Plains accounting/financial modules. Because they'd been using Great Plains for years and this new costing application was the new kid on the block, it was my responsibility to research any differences in the numbers. As a result, I had to quickly learn how to manually query Great Plains tables. Fortunately, I found a fantastic website with valuable information on GP tables: https://victoriayudin.com/gp-tables/gl-tables/.

Version History (Set Preach On)

What color shirt were you wearing on the night of January 16th?

OK, that sounds like an odd question to ask. But here's a different one.

I see from your product price table that product XYZ has a price of $405, and was last updated on 1/16/2020. What was the price before that? And what was the price a year ago?

If the answer is something along the lines of the following, that's sad:

"Well, we'd have to restore a backup"
"We had a developer who suggested that, but that's just too much storage"
"Why? None of our users have ever asked for that"

That last one is particularly sad (and a response I've heard before). If you are waiting for users to suggest the end result of a practice you should know about, you might have made some deeper errors.

> Repeat after me: store version history. Store Version History. STORE VERSION HISTORY!

As recently as in the last five years, I've seen key data in regulated industries that doesn't store version history. Databases have had triggers for decades. Change Data Capture has been in systems for years. And if all else fails, an application can have its down home-grown process for storing version history. I try not to get preachy, but there is just no excuse for this. So repeat after me: store version history. Store Version History. STORE VERSION HISTORY!

This application makes use of version history: The user can view cumulative counts for any day since this crisis began.

Final Thoughts

This concludes my first version of this application, with more to come. Here are some random thoughts to close:

Our World Has Changed, but Good Hasn't

I'm 55 years old, which means I can remember the end of the Vietnam war, the handful of rough economic times, and certainly most CODE Magazine readers are old enough to recall the tragedy of 9/11/2001. I also remember the brief fear in spring 1979 when I lived about 30 minutes from the Three Mile Island nuclear power plant in Pennsylvania.

This situation is far different. Our world has changed **dramatically**. In another year, will our lives go back to the way they were? I have no way of predicting and won't offer any guesses. The old cliché of, "Hope for the best, prepare for and expect the worst" seems as good advice as any.

I've re-learned some personal lessons from this and I'm sure others have as well. One is that small acts of kindness can still go a long way these days. Rod Paddock (CODE editor-in-chief) and I talked on the phone about a month ago. At the time I was quite down. If you've ever talked to Rod, you know that he can lift your spirits. Then about a week ago, he texted me and shared some good personal news he had, which made me feel happy. Again, small acts of kindness...

Get Your Popcorn Ready

No joke: Two to three times a year I'll take some of the more advanced or powerful dashboard applications and cut some 3-5 minute videos that walk through scenarios in the same way the major TV news analysts walk through voting results on election night. It's amazing how many things (little and not so little) I find. Then I'll put those videos into the hands of key business users. It's a technique that can help get users comfortable with your solutions.

Beware of Third-Party Tools

If you've read my prior articles in CODE Magazine or you've attended my presentations, you might remember that I caution people about diving into third party tools. Sometimes they are fantastic. Sometimes they are frustrating or worse. Sometimes you have to do some research to discover that they offer great functionality over the core product, but they are either missing a few key seemingly standard features or suffer from performance problems. I've evaluated several Power BI third-party visualization components over the last year with mixed results. There are some that offer great user experiences that I'd never be able to do quite the same with the regular Power BI visualizations, but also come with some nasty side-effects.

Microsoft offers some nice (and free) add-on visualizations for Power BI on their Marketplace website. I use some for commercial work but have chosen to keep this application free of third-party visualizations, at least for now.

How Can I Run This App?
Where Can I Get the Necessary Files?

Because this application contains data from the public domain (i.e., nothing proprietary), Microsoft Power BI Pro allows me to publish this web app to an open public URL.

The URL is a long one, and because of the chance it could change, you can find it on the opening page of my website, www. KevinSGoff.net. My site also contains the necessary code to embed the application in an iFrame.

As of this writing, I'm manually republishing the data daily. In my next article when I cover enhancements, I'll talk about setting up a data source and ETL process in Azure to do it automatically.

That future article will contain T-SQL code to deal with some allocation of data, and Power BI DAX code to deal with dynamic measures/calculations. My website (www.kevinsgoff.net) has information on the public version of this Power BI application, along with the Power BI Desktop (PBIX) file and other necessary files.

Data Sources

I used the following location on GitHub to pull down daily COVID-19 data. I'm not associated with this data source in any way: I've simply been using it to populate my sets of data for this application: https://github. com/nytimes/covid-19-data

Here are three links that talk further about the source of data.

https://www.nytimes.com/interactive/2020/us/coronavirus-us-cases. html
https://raw. githubusercontent. com/nytimes/covid-19-data/master/us-counties. csv
https://www.nytimes.com/interactive/2020/us/coronavirus-us-cases. html

"Where Do We Go from Here?

I wrote this article to serve as a model (playbook) for creating analytic reporting applications. It's certainly not a complete list. Football teams put new plays in the playbook when they see a need or an opportunity. Additionally, teams win some games without using every play in the playbook, just like specific reporting applications use some of the plays but not others.

In the last year, I've made major changes in my life. I "retired" from all conference speaking and community speaking and from the Microsoft MVP program. My main area of focus is my family, my health, and my full-time job as a database specialist. However, I still plan to continue writing articles here and there, and will look to expand this project. I want to devote an article to incorporating Power BI content into a .NET application, and deploying this project to the cloud. As public sites continue to publish more and more county-wide data, I'll look to incorporate it.

Also, as I look back over this application, there are some features I'd like to add/change, particularly in the trending area and expanding the use of report tooltips. If I could add one additional report page for this article, it would be ranking states and counties by % of increase/decrease in cases/deaths, from the current week going back a week. Although I've included population density, I'd like to add some correlation coefficients to provide mathematical context to how strong density plays a factor in these numbers.

I'm sure others will look at this application and say, "I can think of a better way to do ZYZ." Sometimes I'll look at how I'm presenting data and I'll say to myself, "How I'm doing this is okay, but I know there's a shorter and more elegant path." We all live in the world of Continuous Improvement. Until next time!

Kevin S. Goff
CODE

Talk to an RD

Tiberiu Covaci and Markus Egger

The Microsoft Regional Director Program

The Regional Director Program, or RDs for short, is a program that allows Microsoft to identify influential individuals in an effort to give the community-at-large access to these individuals, and also to provide a point of communication and feedback into Microsoft. Regional Directors do NOT work for Microsoft (and they aren't paid for being part of the RD program), but they have a formal relationship with Microsoft that provides them with great insights and connections within Microsoft.

The Microsoft Regional Director website (https://rd.microsoft.com) defines the RD program in the following words:

"The Regional Director Program provides Microsoft leaders with the customer insights and real-world voices it needs to continue empowering developers and IT professionals with the world's most innovative and impactful tools, services, and solutions. Established in 1993, the program consists of 160 of the world's top technology visionaries chosen specifically for their proven cross-platform expertise, community leadership, and commitment to business results. You will typically find Regional Directors keynoting at top industry events, leading community groups and local initiatives, running technology-focused companies, or consulting on and implementing the latest breakthrough within a multinational corporation."

Regional Directors are expected to have deep technical understanding of many of the Microsoft technologies. Not just that, but RDs are expected to have an understanding of, and experience with, competing technologies. RDs are also expected to go beyond technical expertise and have considerable business expertise. Many RDs present at corporate events, advise governments and NGOs, and many similar scenarios.

Feel free to contact any of the Regional Directors to get access to an RD's expertise and a well-informed opinion that isn't shaped or influenced by having to go along with Microsoft's marketing speak. You can contact RDs to get advice and opinions on your projects regarding technical needs. You can contact them to help analyze how technology will influence your business. You can invite RDs to speak to your project stakeholders, board of directors, or corporate event. And you can contact your RD for many more scenarios.

To continue our "Talk to an RD" column, Markus Egger and Tiberiu Covaci were supposed to get together at this year's RD and MVP Summit at Microsoft in Redmond. Those plans had to be changed in a hurry due to the current Coronavirus Crisis, but that didn't deter these two RDs from chatting about Azure and other topics.

Markus and Tiberiu have been acquainted—even friends—for quite some time. Tibi (as Markus likes to call him) has long been involved with various communities and has organized various events, such as DevSum in Sweden. He even presented some of CODE Magazine's State of .NET Events. (Side note: We've moved our State of .NET events online for the time being. Live events will return once that's possible, but for now, there's a free monthly State of .NET event online, each focusing on a specific topic. Find out more at codemag.com/stateofdotnet). It's therefore no surprise that these two RDs have a lot of interests in common, especially when it comes to cloud computing. Let's listen in!

Markus: Hello Tibi, good to see you!

Tiberiu: Good to see you too!

Markus: I know you usually travel the world, and you've worked and lived in all kinds of places. Where are you right now?

Tiberiu: I'm in Sweden. But yes, for the past 10 years, we've moved all over the world. Now we're back in Sweden, mainly because we have a baby daughter now.

Markus: Congratulations! That's very cool.

Tiberiu: Thank you.

Markus: I even met her last time we got together. It must have been a little over a year ago, I think.

Tiberiu: Yes. She is a year and three months and she keeps us on our tippy-toes. She started to run around the house. So she's the boss for sure. She's clearly making all the decisions in the house at this point.

Markus: That's good. (laughs)

Tiberiu: Yeah, exactly. (laughs)

Markus: We wanted to talk about Azure. You're very involved in Azure. You are a Regional Director, obviously, which is why we're talking, although I'm catching you on the tail end of being a Regional Director, as you're going to move on and join the "mothership" by going to work for Microsoft. From what I understand, you're moving into a role in the Azure team. Tell me about your involvement with Azure. I know you've been passionate about that for a long time.

Tiberiu: I've known about Azure ever since Microsoft started talking about it the very first time. Even before the PDC when they showed the very first data center in a container. I don't know if you remember. Now the show is called BUILD. It was in Los Angeles. I think it was 2009 or 2010 or something like that.

Markus: Yes, over 10 years ago now.

Tiberiu: Yes. A lot of years ago. And they were showing how they wanted to go to the cloud. And my very first thought about Azure was "nah... it's just Microsoft going into the data center business. I don't think they have a chance." I thought they would kill all their partners. You know, all the ISVs and all the other partners that are offering data centers. But then, I actually started to look at what they were offering. And I liked their approach, because, while other companies were starting from the infrastructure-as-a-service (IaaS) offering, Microsoft was thinking, "how can we solve the distributed computing problem?" and "How can we solve what kind of applications our customers are using?"

And now Microsoft, being the enterprise-friendly company that they are, I thought they'd have enough data to understand what the enterprise customer wants. They weren't so much looking into startups or into private people. But they were looking at how can they offer a cloud that is sustainable for an enterprise. So the evolution of it I really liked, because they had Web applications, databases, inter-process communication, that sort of thing. There were problems, but they had good solutions for that. And then we started giving feedback. In a way, we were the unhappy ones all the time. "Oh, but we need access to the machine

that runs my application." So there were always these kinds of things and Microsoft said, "okay, we're opening RDP for you" or "we're opening SSH for you." Then, "we're giving you a virtual machine so you can put whatever you want." So things progressed in a very iterative fashion.

And then they realized that those are good findings, and, based on that, they decided to rewrite the whole thing. And I think that was one of the best decisions ever made by a software company to say, "whatever we've done, we will scrap, and decide to start over with a clean slate." Nobody else does that. You know, I always advocated for something like that because the first version of every software application should be your playground. That's how you learn when you start writing something completely different. In that case, for the second version, you shouldn't patch the first one. You should just start with a clean slate. And they had the guts to do it, you know? And that coming from a company that wanted to keep compatibility with Windows 3.11 for a very long time. (laughs) That says a lot about how much Microsoft changed and how many things they did. So I came to actually like Azure a lot.

These days, one of the things I'm looking at is how developers are doing [on Azure], because I used to be a developer 100% of my time. I'm still doing it.

Markus: As of now, we'll still count you as a developer. Maybe next month when you're officially a Microsoft employee, we'll change our tune. (laughs)

Tiberiu: Yes. (laughs) Just this weekend I was in a hackathon for two straight days and I was working with a few people. We just created a peer-to-peer video communication platform. We had a central server to start the process and then once I started to talk with you, there's no server involved. So there's no bottleneck. There's no problem with infrastructure. It took me about, oh, I'd say, half an hour to deploy that to Azure. So the guy that did the application was running it on Heroku and I just said "okay, let's see how it works." And then my whole weekend went toward doing the whole Web pack and trying to do automatic deployments. Now, whenever I'm committing something, I automatically get the back-end application deployed. And that's a Node.js application. And the front-end deployed on a storage account, with a CDN [content delivery network]. And all that now happens by pushing to a Git branch.

Markus: What did you use as the underlying tech for the actual video stream?

Tiberiu: Web sockets. WebRTC actually. That's what one of the people involved in the hackathon chose to build the application. He's the creator of XSockets. He did that as a technology to be able to use sockets when Web sockets wasn't available everywhere. It's very nice. And that's the cool part. We wanted to have screen-sharing. You just open a new media stream. And then we said, "oh, we want to do file sharing just on that the media stream" and you send the file. It's very cool. That's the power of these technologies, platforms, and services we all now have available.

Markus: It sounds like a magazine article right there actually.

Tiberiu: That would be a good CODE Magazine article, yes.

Markus: You and I have worked together a lot as RDs. We don't have a formal relationship as such, but we're working together on various projects. You've even done a number of State of .NET events for us over in Europe. [State of .NET events are a series of free events sponsored by CODE Magazine. Recently, State of .NET has moved online with monthly free events. Find out more at codemag.com/stateofdotnet]. I know you've always done a lot of Azure content in those State of .NET events as well as other presentations. For a while, you've done a "five things everyone should know about Azure" series that evolved over time. Tell me a little bit about that. What are the latest five things everybody should know about Azure?

In this feature, we eavesdrop on a conversation between Tiberiu Covaci and Markus Egger, both seasoned Regional Directors.

Talk to **Tiberiu** about: Cloud Computing, Azure, Software Development, IoT, DevOps, and many other topics.

Talk to **Markus** about: machine learning and AI, ASP.NET, HTML apps (including Angular, Vue.js, etc.), the cloud, Azure, Microservices, Windows applications, software strategy, and much more.

You can contact him at markus@eps-software.com

For over 20 years **Tiberiu** dedicated his life to improve the quality of the software written by other people by helping them to understand the technological choices they are presented with. His efforts include training, mentoring, advising, motivating, and working side-by-side with their development team. He helped many of his partners and customers to choose the right technological solution for the right problem.

For his passion and dedication, Microsoft has presented Tibi with the Most Valuable Professional and the Microsoft Regional Director awards several years in a row. Since this conversation, he has joined Microsoft as a full-time employee.

Markus Egger is not just an RD, but as the founder and publisher of CODE Magazine, he's also directly associated with this publication. In his main job, he is the President and Chief Software Architect of EPS Software Corp. (a company better known for its CODE brands such as CODE Consulting, CODE Training, CODE Staffing, and CODE Magazine). He is also the founder of other business ventures, such as Tower 48, Wikinome, and others. In his own words, he spends his time "like most software developers, writing production code" both for consulting and custom app-dev customers, and also for his own companies. He has worked for some of the largest companies, including many Fortune 500 companies, such as Microsoft. Markus often takes on the role as a "CTO for hire."

Tiberiu: For the first two or three, it's relatively easy and little changes. You need to know about Web applications, databases, and storage. That's a very good starting point. And you know, when I start talking to people, the first thing they always want to know is "what would be the advantage?" That's always where it starts. When you move to Platform as a Service [PaaS], you shift the responsibility away from taking care of the underlying platform and you can concentrate on your application. There is some—let's call it "hygiene"—that's involved. You need to do some things a little bit differently, too. Like you don't have 100% control over all the things, but I would say in 99% of the cases, you don't really need that. There are very few edge cases where you really need to have control over those small things. Otherwise, there shouldn't be any need for that.

You start with an application and then you get benefits such as scaling. You get custom domains automatically. You get back up as part of it. Everything in your application is taken care of for you without you ever having to worry about it again, and that's a pretty big benefit, even for something as simple as a Web app. You can concentrate on deploying your application and that's it. And as you then start scaling your app, you get benefits, such as affinity to a certain machine being built in. If you scale to ten machines, or to two for that matter, one of the biggest challenges is always to make sure you can still have communications happening, especially if you use session cookies and stuff like that. Because you want to make sure that you talk with the same machine in those situations. Azure solves that problem for you and you never even have to worry about it in the first place. In a sense, it's as if this problem never existed.

Then, the second part is a database. As a developer, I just need to save the data somewhere. Why would I worry about creating the database? Why would I worry about issues such as a back-up? And moving the tapes of my back-up and all those headaches. In Azure, I can just say that I need the database and everything just works out of the box. And it doesn't cost a lot of money. The cheapest production database you can buy in Azure is about $15 per month, which is enough for a medium-sized application. And if you need more, you can just go and say, "okay, I want to pay more for a bigger and more powerful database because I have more customers." It's a simple turn of a dial.

The third part was the storage. Most of the assets that you're using in your Web application are static files. Why would you put the pressure on your web application when you can leverage a ready-made storage solution that does exactly that? Then, there are a number of things you can configure on top of a storage account. For instance, you can add a CDN on top of a storage account. You can even do things such as making sure storage happens in the right place geographically, either for legal reasons, or because you want to store close to the user for best performance. Those are all benefits that would be very difficult to reproduce without a large cloud infrastructure.

Markus: Subscribers of CODE Magazine see this in action, whether they know it or not, because digital CODE Magazine subscription files are served up by Azure storage. This hugely simplified things for us when we moved to that setup.

Tiberiu: Yes, that's a perfect example!

Those are the three of the five services that I was talking about. Then things change a bit more as Azure evolves. We have stuff like Azure Functions. Functions are essentially a piece of code you need to run when something happens.

Markus: Like something that runs on a schedule, or something that runs as a response to an HTTP request, which could even be a different way to implement a REST API call.

Tiberiu: Exactly! Why would you have to worry about creating the virtual machine and having to pay for a virtual machine, when the only thing you should have to care about is your code running when X happens. You deploy your code, and you say to Azure "please do that for me." Whenever you have these events, Azure spins up the virtual machine or a container. It doesn't really matter what it spins up. All you care about is that your code responds and does its thing.

Markus: The whole point is that you don't have to worry what the server is. We refer to it as "serverless." It's a little bit of a misnomer in that sense. There certainly is a server. It's just that you don't care.

Tiberiu: Yeah, exactly. It should be called "I couldn't care less about the server." I think that would have been a better name than "serverless," but they never ask us about such things. (laughs) In the end, you have your code running as you would expect. And scalability happens automatically. You just pay for the time your code is actually running. The more requests, the more machines might be spun up for you. That's the price of doing the business. And at the end of the day, it's still cheaper than running a virtual machine all the time just in case something needs to run or something needs to be done.

Markus: Not to mention that Azure Functions provide a lot of extra features that would be difficult to create on your own. For instance, if your code is supposed to run reliably in response to an incoming email or text message, that, in itself, would otherwise be a serious development and quality control task.

Tiberiu: The last but not the least service that I'm talking about is Azure Key Vault. Because most of the time you need secrets. Think about it like this: If you want to configure your Web application to work with your SQL Server database, you need a connection string to get to it. You need to store a password somewhere. Often, that's done in clear text where, at the very least, it's visible to the administrator. That administrator can put it in the Web application, but those are credentials that the administrator should never get access to. The only one who needs access to those credentials should be your application. With Azure Key Vault, I can configure my Web application as a security principle and that Web application accesses the Key Vault. I don't have any administrator involved who needs to see a database. Of course, I can use the Key Vault for other things, but this is a perfect example and a very common usage scenario.

Now, doing the presentation, which I've done many times, there's one question that keeps coming up: "Do you really have to do everything manually?" That's because in doing the demonstration, I have to show every step and do it manually. I can, of course, automate things, but then I don't think it has the same effect for demonstration purposes. At that point, the overview of five services become six because now you have Azure DevOps you can use, and now also GitHub Actions. I really love that, the simplicity of these services. In an automated fashion, you can create the template that deploys everything that you need. You can have an Azure DevOps pipeline created. It can take the code and it can do the build. It can copy the deployment. You can do whatever you need to do transformations, such as minification. Whatever you need to have done, it's done in a scripted, automated fashion.

Using Azure DevOps, you can even use features such as deployment slots. I can say my dev branch will always do a deployment on my dev environment. And the dev environment is more or less the same as my production environment. Then I have the possibility to run my tests on there.

I have the possibility to do some integration tests and if I see that everything works, I can put it on staging. From staging, I can get the people to try it out. And then I use a slot and my live application is upgraded to a newer version right away.

There are a lot of things like that that will keep developers relevant. Because if you just ignore them, at some point you will have to do the task anyway. If you want your application to really use the power of the cloud, you need to start using at least some of those services.

Markus: It's a very interesting point to think about. How do you get started? Or perhaps "started" is the wrong word, but how do you move to that? We just had, as you know, a very interesting scenario that we'll be publishing articles about in the future here in the magazine, because we had a major ransomware attack. There's a lot of interesting stuff going on, starting with the fact that everything that we had on Azure was unaffected. It was a reversal of what people usually worry about. Normally they think "oh, I gotta be careful with security in the cloud." It was totally the opposite for us, because none of our cloud infrastructure was affected at all. And neither were the people working from home or off-site offices. As a result of all that, we decided to remove the other half of our infrastructure that we still had in our local data center. We decided to recover by moving it all to Azure. It was a wild two weeks because it wasn't a well-planned move in that sense. But it did work out pretty well in the end. I was surprised.

I know you were doing a lot of stuff with getting people to start moving into Azure and move existing infrastructure, right?

Tiberiu: Yes. I don't know if you've seen that, but especially now, with the current situation [the COVID-19 crisis]. People always ask, "what triggered you to move to the cloud?" You know, like the CIO or the CTO who's worried about COVID-19. In the past, people always thought you needed to be in the office. Now, all of that has changed. We see a surge of people who say, "what I need is for my employees to be able to open their laptop in a Starbucks and start working from anywhere." That's the new normal. More and more people have started thinking along those lines. Perhaps they aren't thinking about the Starbuck scenario, but they are thinking about working from home. Either way, if you do the one, the other will just follow. I don't think there'll be any difference because if you know it's safe to work from off-site, then it's as safe or as unsafe at someone's home as it is at Starbucks. I don't think there is too much of a difference.

Markus: I agree. I really think that more acceptance of working off-site will be a lasting change. When we had the 2008 financial crisis, we squeezed out a lot of efficiency from our business processes. Perhaps we've relapsed on some of those, but all in all, what will be the optimizations we implement to overcome the current crisis? It gets harder and harder to think about inefficiencies. One of the things I can think of is to eliminate the overhead people run into that is related to work, but inefficient. Things like hours of commuting to and from work are a huge waste. I see a lot of potential for becoming more competitive by having people dedicate more productive time to work while at the same time reducing the overall hours they spend on work and work-related tasks.

We always think of people working 40 hours a week. But when you count the commute to and from work, and you count that they were somewhere for lunch that was really just because they were at work in the office, and that they had to drive to and from day-care because they can't take care of their own children, and so on, then we probably get to something more like 60 hours a week. I think it would be better to reduce that to 50 hours but spend those 50 hours more productively.

Tiberiu: That's exactly the idea! How can you enable people to work with the things that they need to do? Microsoft did a lot of work around that lately.

Markus: How do businesses move toward that? What do you tell businesses that want to take their first step toward the cloud?

Tiberiu: Microsoft now has something called the Cloud Adoption Framework. You can think of it as a DevOps approach to moving to the cloud. If you want to start that process, Microsoft has a DevOps generator that will give you a starting point. You'll get all the activities with all the features and user stories and so on. You can have your Kanban board that includes things such as "you need to convince stakeholders," "you need to address those business people in order make certain decisions," and so on. One of the things I love about that is that most companies think their needs are unique. And they are all different, of course. But when you move infrastructure to the cloud, there are certain steps almost every company has to take. This framework really helps with that.

I see a lot of the same patterns in many cases. A lot of customers say "we want to start with Software as a Service and if we can't find anything, we want to write our own using Platform as a Service. If we don't find that, then let's use Infrastructure as a Service." Guess what? 80% of people moving to the cloud start out moving to Infrastructure as a Service in spite of what the CEO and CTO decided. It happens on its own. Most people are in a rush, so they start moving the machines they have to the cloud as they are. The explanation is pretty simple. Most companies just don't have the time or means to do a full analysis to see if they can move to Platform as a Service.

Markus: We've had the same scenario. The hardest part for us to move after the ransomware attack we suffered was to move the database. We had a very, very large database, which wasn't really designed for the cloud but needed to get up and running. We had two really tough challenges. One was how to get this much data to the cloud. For us, there was little choice, because we were dead in the water. We uploaded to the cloud and however long it took, it took. Then, the second aspect was that it just wasn't designed for the cloud. We had to change some of our data structures. Not much, as it turned out. In the end, it was relatively easy. As a first step, we just spun up a VM and put it in there, as if it were its own SQL server. Then we changed data structures a little bit, and then broke the database apart a little bit, and cleaned it up a little bit while we were at it. And now it's fully on SQL Azure as a SaaS platform.

Tiberiu: Cool.

Markus: What do you see as the hardest part? Is it usually data? Or do you just see old apps that need to change?

Tiberiu: It's people. Still people. That's the biggest challenge I see with every company that wants to move to the cloud. Regardless of where the decision comes from, it's the people that resist the most. They make it so much harder in most situations. They don't want to change. I think that's part of human nature. What I've also seen is that a lot of them are very scared of change because they don't know what to expect. It's completely unknown territory and it's something completely different.

I think for me, at the beginning, it was pretty hard as well because the terminology is completely different. You need to get used to that. But then, when you start explaining, they get the hang of it pretty quickly. In Azure, unlike in conventional data centers, it's not about hardware anymore. Everything is defined by software. At the end of the day, things

are still the same, at least from an operational standpoint. Of course, if you talk about monitoring, if you talk about networking, and so on, there are other tools that you need to put in place. They're still the same kinds of ideas. Then, of course, you talk with people, and they still think, "oh, I need my DMZ." And I'm like, "are you sure that's what you need?" because I'm not really sure.

Markus: You're applying your old concepts to a new setup. And some of those concepts may just not apply anymore.

Tiberiu: They used to work with those concepts and that's why they want to keep it like that. That's the biggest challenge. I don't think it's a technological one. It's people.

Another problem that I see quite often is that they're afraid that if they move a machine, they don't know what all the dependencies are. Most big organizations have a mess of IT. I think there are very few where that isn't the case. There are very few companies that have a complete list of not just all their servers and what's on them, but also what accesses them and what might break if they're moved. There are tools out there that can help with that. They can detect how the machines are working. They can give you a fair picture about what you have. But still, companies are often in a situation where they don't really have everything they need. In most cases, you need to make sure that when you move something, you're able to move back quickly when you discover that you broke something.

Markus: Which is pretty much what the case was for us. I mean, obviously, because our ransomware attack wiped out every single server we had, and because we had to preserve that for forensic analysis, we had no choice. We had to move forward. It was challenging. Getting the major pieces up and running was not so difficult. We had our external facing things, like websites and so on, up and running relatively quickly. But then it was the little internal details that you just didn't even remember, that were the hard parts. Although we had some pretty good docs overall, we didn't have all those details well-documented. Like the little WinForms tool that somebody wrote themselves 15 years ago that was still in use and had access to the database in some way. And you just forgot about that and now it didn't work anymore, and there was really only one person in the company that even still used that.

But I also have to say, when we looked at this maybe two or three years ago, when we started our move to Azure, we identified that about 50% of our infrastructure was easy to move and we moved it relatively quickly. The other 50% was a different story. And it isn't even that we knew it was going to be hard to move. It's more like we didn't know how hard it was going to be. We had a hard time assessing what pieces there are that we don't really know about or that you forget about or that aren't really worth moving. Due to our attack, we were forced to make those changes and take the plunge. And I'm glad, in a way, that we did because we're probably better off today. But we went through a few rough weeks to make that all work.

You know, one of the biggest drivers today is that people are at the end of their contract cycle with their current data center. Or maybe they need to buy new hardware and decide to move to the cloud instead. So three or five years have gone by and they need to switch to new machines, or they push to the cloud instead.

The very first cloud customer I had I actually managed to scare away. (laughs) They decided to move to another data center, not to Azure. I think that that was my mistake. I went to them and showed them as much of Azure as possible in all its beauty. I think to them, it was overwhelming. They just wanted to move some servers and some websites. And I'm like "wait a second. If you do that, you can use the service directly," and then all of a sudden, it was too much for them. So they decided to just move to a local data center. That was a hard lesson to learn, because it was a project on the order of 20 million pounds by the time it was all said and done. So that was a bit of an eye opener for me, because I thought that if I like it, everybody will like it, especially if you talked nicely about it. But people were still hung up from their own technology and their own way of thinking. And it's easy to overwhelm them with everything Azure can do.

Tiberiu: I hear that.

Markus: "Eye opening" is a good term. It was really eye opening for us to have a scenario where we needed to do this for ourselves. Usually, we do it for customers and help them; when it's their pain, that doesn't seem so bad, right? But when all of a sudden it's your own organization, it truly is an eye opener. And that's good, because it really helped us in supporting our customers better.

Hopefully you will still be able to do that. I know in your new role at Microsoft you're not going to be working directly with customers in a consulting fashion anymore, but you're still doing the same thing for Microsoft essentially when you join them, right?

Tiberiu: One of the things that I will have to do is not only excite developers but more of a general audience of stakeholders. It might involve me talking to the customer. Maybe not doing the groundwork but advising them on how to move. At least having initial discussions and trying to understand what they need and maybe explaining to them how I can help. It will still be an interesting position. So I'll see.

Markus: Excellent. I wish you the best of luck with all that. It sounds pretty exciting.

It's getting late where you are. I'm just getting started with my day here. This is definitely the most geographically remote RD Talk I've ever done. We are pretty much exactly on opposite ends of the planet, with a 12-hour time difference. That in itself is very cool and shows how far we've come. It's been great talking to you.

Tiberiu: Likewise.

Markus: I'm sure the times will come again when we can all meet in person. Maybe at one of the conferences you were involved with or, or at Microsoft.

Tiberiu: I'll still do my conference. [Tibi organizes DevSum in Sweden] We are going to do it digitally this year. There are about 15 to 17 speakers from Sweden. Most of them are from Stockholm. We also have some international speakers. Tim Huckaby, our fellow RD [and former participant in this column] is going to do our keynote remotely.

Markus: It's great that you're still doing it. Keep pushing forward!

Tiberiu: Yeah, exactly. We have to.

Markus: So is Microsoft with BUILD online and other online events. There will be different kinds of events, but I think there will also be interesting ideas coming out of those.

Tiberiu: That's right. Microsoft has announced they're moving all their events online between now and the summer of 2021.

Markus: I hope I will see you sooner than that. And best of luck with your new role within Microsoft.

Tiberiu: Thank you.

Markus Egger

Using a Scripting Language to Develop Native Windows WPF GUI Apps

In this article, I'm going to talk about developing WPF (Windows Presentation Foundation) GUI using a scripting language. Why would we use a scripting language for making a graphical interface? There are several benefits:

Vassili Kaplan
VassiliK@gmail.com

Vassili is a former Microsoft Lync developer. He's been studying and working in a few countries, such as Russia, Mexico, the USA, and Switzerland.

He has a Masters in Applied Mathematics with Specialization in Computational Sciences from Purdue University, West Lafayette, Indiana and a Bachelor in Applied Mathematics from ITAM, Mexico City.

In his spare time, Vassili works on the CSCS scripting language. His other hobbies are traveling, biking, badminton, and enjoying a glass of a good red wine.

You can contact him through his website: http://www.iLanguage.ch or e-mail: vassilik@gmail.com

- The GUI functionality could be changed at runtime, without recompiling the binary.
- The GUI can have a different customization for the same binary.
- Programming can be done much quicker: C# implementation that may require many lines of code, can be embedded in just one or two scripting language statements.

As a scripting language, I'm going to use CSCS (Customized Scripting in C#). This is an easy and a lightweight open-source language that has been described in previous CODE Magazine articles: https://www.codemag.com/article/1607081 introduced it, https://www.codemag.com/article/1711081 showed how you can use it on top of Xamarin to create cross-platform native mobile apps, and https://www.codemag.com/article/1903081 showed how you can use it for Unity programming. CSCS resembles JavaScript with a few differences, e.g., the variable and function names are case-insensitive.

> A picture is worth a thousand words. An interface is worth a thousand pictures.
>
> *Ben Shneiderman*

You're going to see how to integrate scripting into a WPF App and use it for different tasks:

- Creating new GUI widgets
- Moving/showing/hiding existing widgets
- Dynamically modifying GUI
- Providing some non/GUI functionality, e.g., working with SQL Server
- Creating new windows from XAML files on the fly
- Automatic binding of the Windows and Widget events

Using a scripting approach significantly decreases development time. What could've taken many lines of code in C#, takes just one command in CSCS, or even nothing at all. For example, there's no need to bind a widget or a window event to the event handler. This binding is created by the CSCS Framework behind the scenes. The developer just needs to fill out the corresponding function, which will have an empty body by default. You'll see some examples of that later on.

Also, it takes much less code to access the SQL Server, to get data from it, and to update existing SQL tables using CSCS than C#. The CSCS Framework is open source and you're free to use and modify it as you wish. The framework GitHub link is provided in the side bar.

You can add some missing events or widgets there in the same way it's done with other widgets. Similarly, you can add other databases, like Oracle, or add any missing database functionality. Let's start with setting up a sample project.

Setting Up Scripting in a WPF Project

Note that I used Visual Studio 2019 Express Edition for all of the examples in this article.

To set up CSCS scripting in a WPF project, first open a new WPF project from Visual Studio [by selecting File > New Project... > WPF App (.NET Framework)]. After that, you can add a CSCS framework manually to the project. The CSCS framework is open source and can be downloaded from https://github.com/vassilych/cscs.

Alternatively, you can download an existing WPF sample project from https://github.com/vassilych/cscs_wpf with everything already set up, and start playing with it by changing the GUI, some parameters, etc. It also has a few sample scripts.

After that, your Visual Studio main view will look similar to the one shown in **Figure 1**.

How Scripting is Triggered in a WPF Project

The code that initializes the CSCS scripting engine and starts it up is in the MainWindow.xaml.cs file, in the MainView_Loaded() method. As its name suggests, this method is triggered after the main view is loaded; then you can do some GUI adjustments, subscribe to the GUI events, etc. Here is this method:

```
void MainView_Loaded(object sender,
            RoutedEventArgs e) {
  CSCS_SQL.Init();
  CSCS_GUI.MainWindow = this;

  var res = this.Resources;
  var cscsScript = (string)res["CSCS"];

  Console.WriteLine("Running CSCS script: " +
    cscsScript);
  CSCS_GUI.RunScript(cscsScript);
}
```

The name of the CSCS scripting file to run is taken from the resources. Specifically, it's defined in the MainWindow.xaml file. Here's an example of how it can be defined:

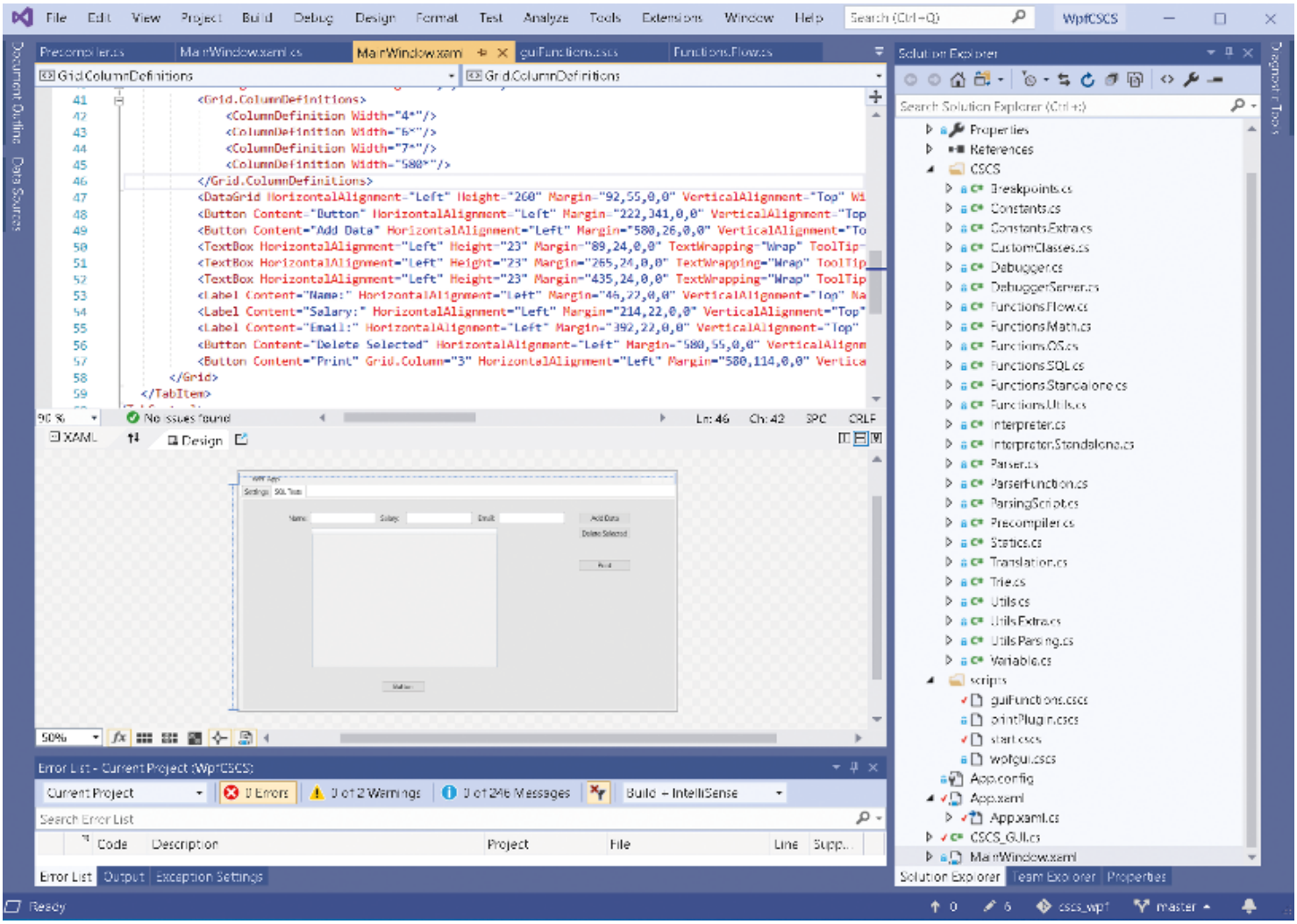


Figure 1: Microsoft Visual Studio 2019 WPF Project with integrated CSCS Scripting

```
<Window.Resources>
  <sys:String x:Key="CSCS">
  ../../scripts/start.cscs</sys:String>
</Window.Resources>
```

CSCS scripts are first-class citizens of the WPF projects and are included in the scripts folder on the top level (see the right side of **Figure 1** for details).

The contents of the start.cscs script is rather straightforward—it just says what script to run:

```
StartDebugger();
include("wpfgui.cscs");
```

The StartDebugger() statement is optional. It starts the scripting debugger so that you can connect and debug scripting from Visual Studio Code. A previous CODE Magazine article explains its usage: https://www.codemag.com/Article/1809051. Basically, you need to download a Visual Studio Code CSCS Debugger and REPL extension from the marketplace.

The real action happens in the wpfgui.cscs file. This way, you can change what file to include in the start.cscs file if you work with different projects.

General Structure of a WPF Project with Scripting

The general structure of a WPF project with scripting is similar to a plain vanilla WPF project with this exception: You create the GUI in mainView.xml as usual, but you don't create any event handling. All this will be handled by the CSCS GUI module. This GUI module is part of the CSCS project and it's called CSCS_GUI.cs

The GUI widgets can also be created purely in CSCS and you'll see an example of this later on.

It's a good idea to take a look at the CSCS_GUI.cs file at the GitHub location mentioned on the sidebar. You will see how the C# widget events are hooked to the CSCS methods.

It works as follows: First you define a widget in MainWindow.xml. The relevant parameter is DataContext in the widget definition. For instance, if a new button is defined as following:

```
<Button Content="Open File" HorizontalAlignment="Left"
Width="75" Name="button1" DataContext="myButton"/>
```

Then a few event handlers in CSCS are created automatically by the CSCS GUI module. Some of these event handlers are the following:

- myButton@click: This function will be triggered when the user clicks on the button.
- myButton@preclick: This function will be triggered when the user presses the mouse on the button.
- myButton@postclick: This function will be triggered when the user releases the mouse on the button.
- myButton@mouseHover: This function will be triggered when the user's mouse is over the button.

> The Event Handlers will be created automatically, e.g., if you define myButton widget, myButton@Click will be triggered as soon as the user clicks on myButton.

For a text field, there are different event handlers. For instance, if you have this definition of a text field:

```
<TextBox HorizontalAlignment="Left"
 Width="75" Name="textfield1"
 DataContext="myTextField"/>
```

Some of the event handlers created by the CSCS GUI module are going to be:

- myTextField@textChange: This function will be triggered when the user types anything in the text field.
- myTextField@keyDown: This function will be triggered when the user presses any key.
- myTextField@keyUp: This function will be triggered when the user releases any key.

For a combo-box (a drop-down widget), a typical automatically created event handler is:

- myCombobox@selectionChanged: This function will be triggered when the user selects an entry.

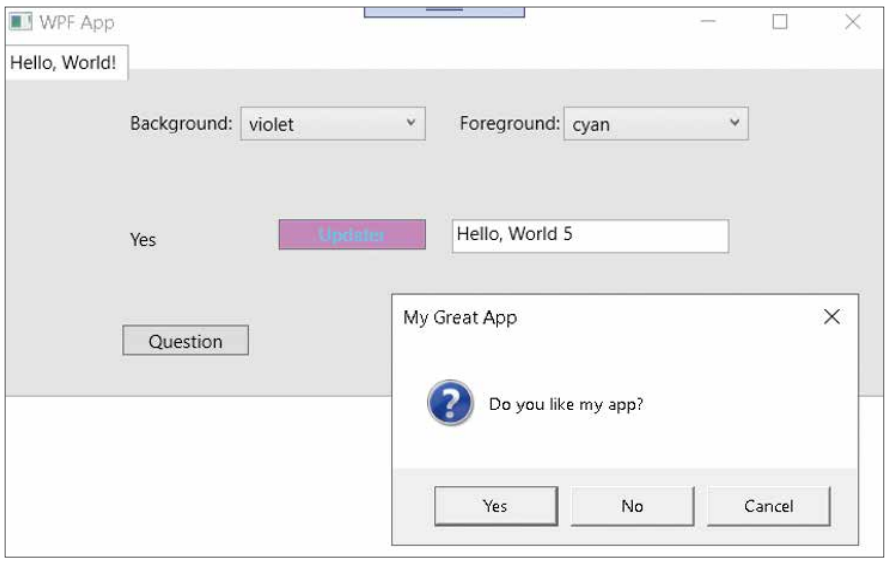


Figure 2: WPF Hello, World! With Scripting

Even though the event handlers will be created by the CSCS framework, the function bodies, by default, will be empty. In the next section, you're going to see how these event-handling functions can be filled with some useful stuff in CSCS code.

Hello World in WPF with Scripting

Let's take a look at a relatively simple GUI example. Let's create a GUI in Visual Studio by dragging and dropping different widgets, as shown in **Figure 2**.

What you're going to do in CSCS is to add some life to the GUI created in **Figure 2**. First, initialize the global data and populate both combo boxes with data as follows:

```
comboItems = {"white", "green", "red", "yellow",
              "black", "pink", "violet",
              "brown", "blue", "cyan"};
AddWidgetData(comboBGColors, comboItems);
AddWidgetData(comboFGColors, comboItems);
count = 0;
```

Next, add the event handlers:

```
function comboBGColors@SelectionChanged(
         sender, arg) {
  SetBackgroundColor("buttonUpdater", arg);
}
function comboFGColors@SelectionChanged(
         sender, arg) {
  SetForeGroundColor"buttonUpdater", arg);
}
function buttonUpdater@Clicked(
         sender, load) {
  SetText("textArea", "Hello, world " +
          (++count));
}
function buttonQuestion@Clicked(
         sender, load) {
  result = MessageBox("Do you like my app?",
    "My Great App", "YesNoCancel", "question");
  SetText("labelAnswer", result);
}
```

> CSCS is case-insensitive. Therefore, the method signature buttonUpdater@Clicked is the same as Buttonupdater@clicked.

The first two functions are triggered when the combo-box values change. They update the colors of a button. The third function updates the value of a textbox with a global variable counter, being incremented on each click. The fourth function shows a modal message box and waits until the user clicks on one of its buttons. The very moment of this message box shown to the user is shown in **Figure 2**. The name of the button that user clicked on is shown in the labelAnswer widget.

Note that you only need to implement the event handlers; binding them with the actual events will be taken care by the CSCS Framework.

CSCS Function	Description
SQLConnectionString (connString)	Initializes database and sets it to be used with consequent DB statements. The connString is in a standard SQL initialization form. **Example:** SQLConnectionString("Server=myPC\\SQLEXPRESS;Database=CSCS;User Id=sa;Password=sqlpassword;");
SQLTableColumns (tableName)	Returns a list of all column names of a given table. **Example:** cols = SQLTableColumns("Users");
SQLQuery (query)	Performs a select statement and returns an array with the results. **Example:** results = SQLQuery("SELECT TOP 5 * FROM Users where id <= 100");
SQLNonQuery (sqlStatement)	Performs a non-query statement, like insert, delete, etc., and returns the number of records affected. **Example:** SQLNonQuery("Delete from Users where id=10");
SQLInsert (tableName, colNames, data)	Inserts data from the data CSCS array into the table. **Example:** data = {"John", 5000, "john@me.com"}; result = SQLInsert("Users", "Name,Salary,Email", data);
BindSQL (widgetName, tableName)	Binds WPF widget to a given SQL table. After this statement, the widget is going to have the data from the passed table. **Example:** BindSQL("myGrid", "Users"); // myGrid is a WPF DataGrid

Table 1: CSCS Cross-Platform Functions for Mobile Development

That's it—as you can see, there would be much more coding if the same functionality were implemented directly in C#. The whole source code for this example can be consulted on the GitHub (see the links in the sidebar).

Using SQL Server in Scripting

In this section, you're going to see how to use SQL Server together with WPF and scripting. You don't need any extra **include** statements in the scripting. Do make sure that the System.Data.dll is included in the project references (it's already included in the sample project).

Table 1 contains available SQL functions with corresponding examples. You'll also see more examples below.

Let's see an example of using the SQL functions described above. For the example, create the following table in your SQL database:

```
CREATE TABLE [dbo].[Users](
          [Id] [int] IDENTITY(1,1) NOT NULL,
          [Name] [varchar](255) NULL,
          [Salary] [real] NULL,
          [Email] [varchar](255) NULL,
          [Created] [datetime] NOT NULL,
PRIMARY KEY CLUSTERED ([Id] ASC)
       WITH (PAD_INDEX = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Users] ADD DEFAULT
    (getdate()) FOR [Created]
```

Using the SQL statement above, you're making sure that the **Id** filed is autogenerated by incrementing the previous Id and the **Created** filed is autogenerated from the current date.

Then create GUI in Visual Studio, as shown in **Figure 3.** The data table in the middle is a DataGrid. The XAML file that contains this example is available in the sample WPF project on GitHub.

Figure 3 shows the moment after the user clicked on the "Add Data" button to add a new entry but before the data grid has been auto-populated. After adding an entry from the GUI, the SQL table Users will look like **Figure 4**.

Now let's see how this is implemented in CSCS. Let's start with the implementation of the function when the user clicks on the Refresh button, which fills out the table in the GUI in **Figure 3**:

```
function buttonRefresh@Clicked(sender, load) {
  BindSQL("myGrid", "Users");

  query = "SELECT TOP 15 * FROM Users " +
          "where id <= 100";
  print(SQLQuery(query));
}
```

As you can see, it's pretty short. But actually it's even shorter than it looks; to fill out the table, you need just one statement: **BindSQL("myGrid", "Users")**. It will populate the DataGrid with the contents of the Users table, setting the header row with the column names from the SQL Server database defined earlier with the SQLConnectionString() CSCS function.

The other two statements are there just for the illustrative proposes—they show what happens if you want to run a SQL Server select query. The result of that query is printed in the Output Window in Visual Studio. This result is a list consisting of the following entries:

```
{{Id, Name, Salary, Email, Created},
{1, John, 20000, johnny@gmail.com,
    9/10/19 8:11:43 PM},
{2, Juan, 15600, juan@gmail.com,
    9/10/19 8:11:43 PM},
{3, Ivan, 14900, ivan@gmail.com,
    9/10/19 8:11:43 PM},
{41, Johannes, 12345, j@me.com,
    3/19/20 2:05:59 AM}}
```

The first list entry is the header (column names in the corresponding database table) and the rest are the table values (corresponding to the actual values, as shown in **Figure 4**).

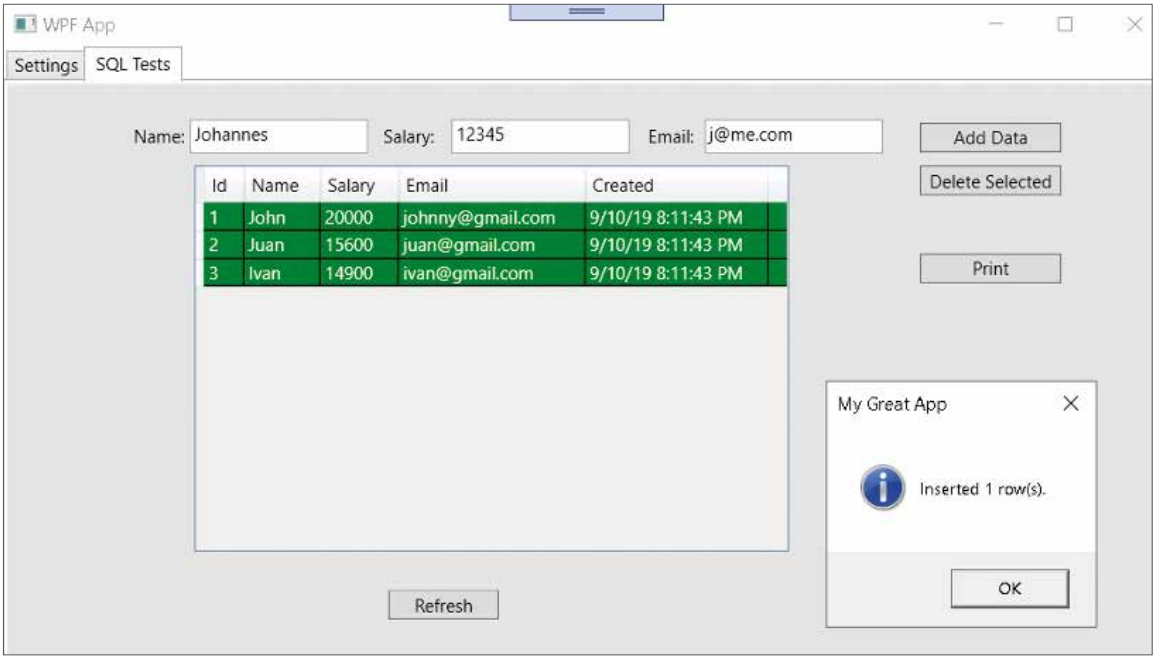


Figure 3: WPF with scripting and SQL Server example

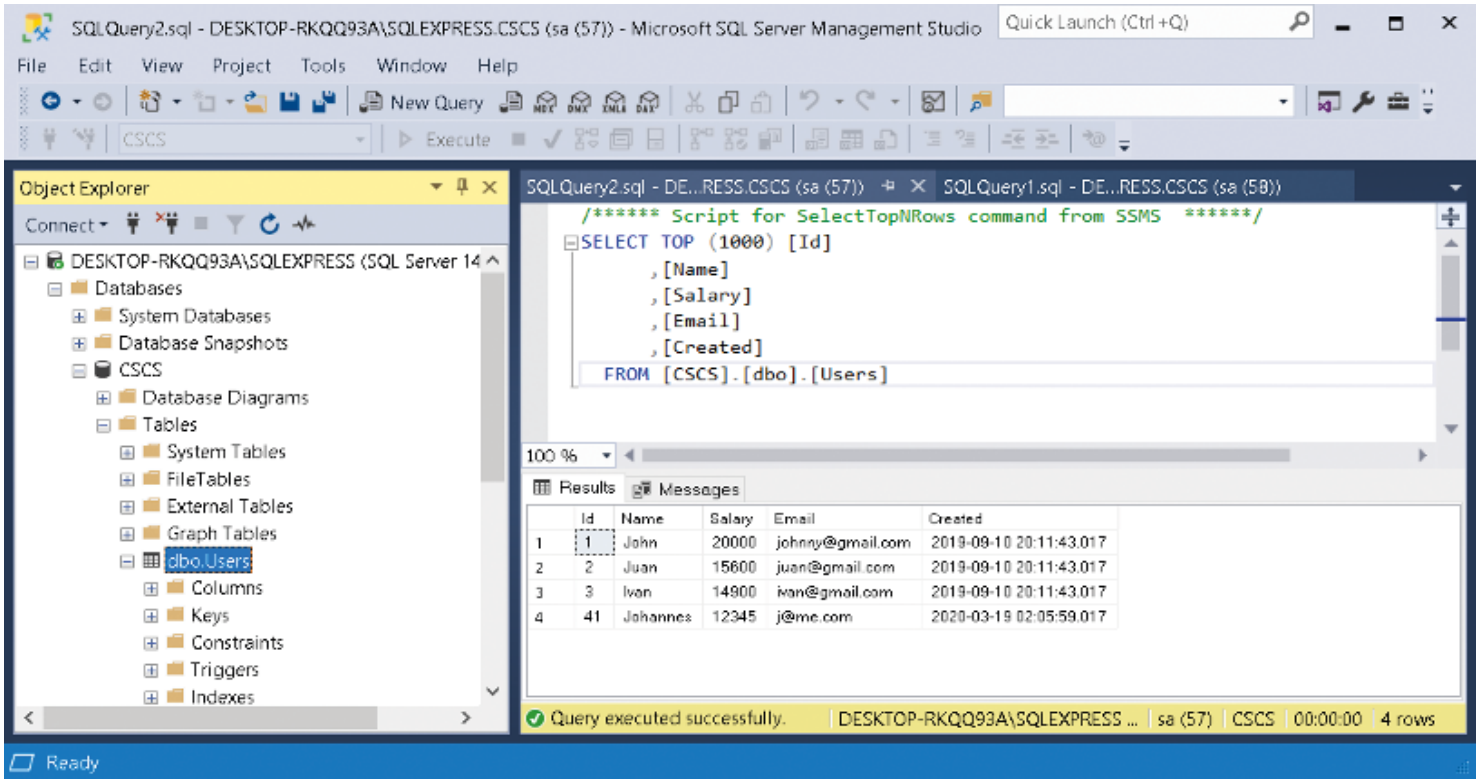


Figure 4: The contents of the Users table in the SQL Server Database

When the user clicks on the Add Data button, the following CSCS function is triggered:

```
function buttonAddData@Clicked(sender, load) {
  data  = {GetText("textName").Trim(),
           GetText("textSalary").Trim(),
           GetText("textEmail").Trim()};
  if (data[0] == "" || data[1] == "" ||
      data[2] == "") {
    MessageBox("Please fill out all fields",
               title, "OK", "warning");
    return;
  }
  try {
    result = SQLInsert("Users",
             "Name,Salary,Email", data);
```

```
    MessageBox("Inserted " + result +
        " rows(s).", title, "OK", "info");
    buttonRefresh@Clicked(sender, "");
  } catch (exc) {
    MessageBox(exc, title, "Cancel", "error");
  }
}
```

This function explains why **Figure 4** has the message box saying that a new entry was inserted even though the DataGrid in the GUI hasn't been updated; this is because the buttonRefresh@Clicked() method is called after that. If you want to change this functionality and update the DataGrid first, simply put the buttonRefresh@Clicked() method before triggering the MessageBox.

Note that the data array containing the rows to be inserted, has only Name, Salary, and Email columns, even though the Users table has two more columns (see **Figure 4**). This is because of the way the table was created: the first column, ID, will be auto-incremented with each insert, and the last column, Created, will automatically contain the current timestamp.

Finally, here is the implementation of the CSCS function triggered when the user clicks on the Delete Data button:

```
function buttonDeleteData@Clicked(sender, load)
{
  selected = GetSelected("myGrid");
  if (selected.Size == 0) {
    MessageBox("Nothing has been selected.",
      title, "OK", "warning");
    return;
  }

  deleted = 0;
  for (row : selected) {
    deleted += SQLNonQuery(
      "Delete from Users where id=" + row);
  }
  MessageBox("Deleted " + deleted + " row(s).",
             title, "OK", "info");
  buttonRefresh@Clicked(sender, "");
}
```

The GetSelected(widget) is a CSCS function that returns a list of all rows being selected. In this example, you iterate over all selected DataGrid rows and run a Delete statement for each of them.

If there are many rows selected, it would make sense to run the Delete statement just once for all of them. In this case, you can slightly modify the code above in the Delete statement by including all the selected rows and then concatenating them with an "or".

Implementation of SQL Functions in C#

Let's see how the SQL functions shown in the previous section are implemented in C#. The shortest one is a SQL nonQuery function:

```
class SQLNonQueryFunction : ParserFunction {
  protected override Variable Evaluate(
        ParsingScript script) {
    var args = script.GetFunctionArgs();
    var query = Utils.GetSafeString(args, 0);

    using (SqlConnection con = new
      SqlConnection(CSCS_SQL.ConnectionString))
    {
      using (SqlCommand cmd =
          new SqlCommand(query, con)) {
        con.Open();
        cmd.ExecuteNonQuery();
      }
    }
    return new Variable(true);
  }
}
```

To bind it with the Parser, register this function in the initialization phase, as follows:

```
ParserFunction.RegisterFunction("SQLNonQuery",
    new SQLNonQueryFunction());
```

That's it. As soon as the parser encounters the "SQLNonQuery" token, it triggers the execution of the Evaluate() method on the SQLNonQueryFunction shown above. The script.GetFunctionArgs() method extracts all of the parameters passed to the CSCS "SQLNonQuery" method.

The implementations of other functions, like SQLQuery(), SQLInsert, etc., are similar. I encourage you to consult the source code and take a look at other implementations as well.

Creating, Showing, Hiding, and Moving Widgets with Scripting

Using the CSCS Scripting language, it's easy to implement most of the GUI related functions. Here's an example of showing and hiding a widget when the user clicks on a button:

```
function myButton@Clicked(sender, load) {
  if (showCounter % 2 == 1) {
    ShowWidget("gridTable");
  } else {
    HideWidget("gridTable");
  }
  showCounter++;
};
```

Similarly, here is a function to add a new button:

```
function addButtons@Clicked(sender, load) {
  name = "newButton" + newWidgets;
  text = "New Button" + (newWidgets++);
  AddWidget(name, "button", text, x, y,
            width, height);
  x += 100;
}
```

The C# implementation of all of these GUI functions is similar. Here is an implementation of a function to show or to hide a widget:

```
class ShowHideWidgetFunction : ParserFunction
{
  bool m_showWidget;
```

```
  public ShowHideWidgetFunction(bool showWidget)
  {
    m_showWidget = showWidget;
  }
  protected override Variable Evaluate(
          ParsingScript script) {
    var args = script.GetFunctionArgs();
    var name = Utils.GetSafeString(args, 0);
    Control widget = CSCS_GUI.GetWidget(name);

    widget.Visibility = m_showWidget ?
       Visibility.Visible : Visibility.Hidden;
    return new Variable(true);
  }
}
```

To register the implementation above with the parser, the following statements must be executed at the start-up time:

```
ParserFunction.RegisterFunction("ShowWidget",
    new ShowHideWidgetFunction(true));
ParserFunction.RegisterFunction("HideWidget",
    new ShowHideWidgetFunction(false));
```

The implementation of other GUI functionality, like removing or moving widgets, is very similar and can be found by consulting the source code.

> To have any C# functionality available in CSCS, first create a class deriving from the ParserFunction class and override its Evaluate() method. Then register the new class with the parser.

Creating New Windows Dynamically from XAML Files

Another useful feature of using a scripting language for GUI development, is the possibility to create new GUI windows at runtime with just a few scripting commands.

For example, you can add any new window to an existing application, if you have this window sourced as a standard WPF XAML file. This XAML file can be created using Visual Studio or any text editor. Note that the XAML file doesn't have to be compiled into the executing program but can be added dynamically on the fly.

Here is an example of such a XAML file containing a window definition:

```
<Window xmlns:local="clr-namespace:WpfCSCS"
  Title="Window1" Height="350" Width="500">
  <Grid Margin = "0,0,230.6,172" >
  <Button Content="Button1"
    HorizontalAlignment= "Left"
    Margin="48,68,0,0"
    VerticalAlignment="Top" Width="75"
    DataContext="Window1Button"/>
  <Label Content = "Window1"
    HorizontalAlignment="Left"
    Margin="48,10,0,0"
    VerticalAlignment="Top"
    Height="44" Width="76"/>
  </Grid>
</Window>
```

This window can be added either as a modal window (i.e., having a parent window) or a stand-alone window. To add this window as a modal window, execute this CSCS statement:

```
win = ModalWindow(pathToWindowXAMLFile);
```

Optionally, you can provide a parameter indicating the window's parent (by default, it's the main window). To create a stand-alone window from a XAML file execute this CSCS statement:

```
win = CreateWindow(pathToWindowXAMLFile);
```

Both of these functions, ModalWindow() and CreateWindow(), will compile the XAML file on the fly and show user the corresponding Window GUI.

> For each new window dynamically created from a XAML file, the window event handlers will be created automatically.

Once a new window is created, there are a few event handlers bounded to the Window creation and destruction events. These event handlers will be created automatically for each new window created from a XAML file.

For instance, if you want to have some code executed when the user closes a window, implement a function called WindowName_OnClosing() (by default it has an empty body). Here is an example of such function implementation:

```
function Window1_OnClosing(sender, load)
{
  result = MessageBox(
     "Do you want to close this window?",
      title, "YesNo", "info");
  return result != "Yes";
}
```

If this function returns true, the window closing will be canceled.

Similarly, the following functions can be implemented in CSCS:

- WindowName_SourceInitialized (corresponds to the window SourceInitialized event)
- WindowName_Activated (corresponds to the window Activated event)
- WindowName_Loaded (corresponds to the window Loaded event)

References

CSCS Language E-book:
https://www.syncfusion.com/ebooks/implementing-a-custom-language

CSCS WPF Sample Project:
https://github.com/vassilych/cscs_wpf

CSCS Repository:
https://github.com/vassilych/cscs

CSCS Visual Studio Code Debugger Extension:
https://marketplace.visualstudio.com/items?itemName=vassilik.cscs-debugger

Windows Lifetime GUI Events:
https://docs.microsoft.com/en-us/dotnet/framework/wpf/app-development/wpf-windows-overview - window-lifetime-events

- WindowName_ContentRendered (corresponds to the window ContentRendered event)
- WindowName_Closing (corresponds to the window Closing event)
- WindowName_Deactivated (corresponds to the window Deactivated event)
- WindowName_Closed (corresponds to the window Closed event)

A good explanation and overview of the Windows lifetime events mentioned above can be consulted here: https://docs.microsoft.com/en-us/dotnet/framework/wpf/app-development/wpf-windows-overview#window-lifetime-events.

Wrapping Up

You need just two steps to have any C# functionality available in CSCS. First, implement a class deriving from the ParserFunction class and override its Evaluate() method. Second, register the new class with the parser using the ParserFunction.RegisterFunction() method.

In this article, you saw how to use a scripting language to change a WPF GUI at runtime and also how to add muscle to the XAML file created with Microsoft Visual Studio.

Basically, CSCS is a functional language; with just a few CSCS statements, you can achieve something that would have taken you tens or even hundreds of lines of the C# code.

The code shown in this article is an invitation to explore. After reading it, I hope you can extend the GUI functions shown in this article and create the new ones. If it's not available yet in the sample project, you can implement any GUI-related functionality in CSCS similar to what I did in this article.

I'm looking forward to your feedback: Tell me what you create with CSCS and WPF, or what other features in CSCS you would like to see.

Vassili Kaplan
CODE

(Continued from 80)

4. Build Competent Software

This is a phrase I have coined: Competent Software. Competency is about having the necessary things to be successful. For example, to be a competent lawyer, among other things, you need to graduate from law school, qualify, sit for and pass the bar exam, fulfill your annual continuing education requirements, abide by the rules of professional responsibility, and not be suspended/disbarred from the practice of law. That is a nice checklist of items. I have the following checklist of items for Competent Software:

- **Delivery:** Delivery is a feature. Unless people can use and interact with your software, it's useless and of no value.
- **Capable:** Your software must be built for a specific purpose. It must be able to do what is required of it by the business. If it can't, it's useless and of no value.
- **Reliable:** Your software must work. It must be able to handle exceptions and, at the very least, degrade gracefully. If a dependency isn't available, it's okay that your software doesn't work. The question is, "how does your software respond?" Or does it just react with a cryptic error message? If your software can't be depended upon to work, it's useless and of no value.
- **Scalable:** If your software needs to be available to thousands of users, then it must be designed, built, and deployed with that capability. If it isn't, although any single user may find value, to the business that sponsored the software, it's useless and of no value.
- **Verifiable:** It's not enough that your software works. You must be able to demonstrate that it works under friendly and adverse conditions. This's what testing is all about. We also verify through metrics and instrumentation. If you don't test completely and correctly, by log, instrument, etc., you're throwing caution to the wind. Your software might work. Or, you may have a Galloping Gertie on your hands. In many cases, SOX, SOC, FDA, and other regulations require a complete working and documented testing regime. You may have delivered capable, reliable, and scalable software. But if you can't verify it, it will be useless and of no value because it won't see the light of day.

John V. Petersen
CODE

CODE

Jul/Aug 2020
Volume 21 Issue 4

Group Publisher
Markus Egger

Associate Publisher
Rick Strahl

Editor-in-Chief
Rod Paddock

Managing Editor
Ellen Whitney

Content Editor
Melanie Spiller

Editorial Contributors
Otto Dobretsberger
Jim Duffy
Jeff Etter
Mike Yeager

Writers In This Issue
Markus Egger
Vasili Kaplan
Sahil Malik
Paul D. Sheriff
Shawn Wildermuth
Kevin S. Goff
Julie Lerman
John V. Petersen
Helen Wall

Technical Reviewers
Markus Egger
Rod Paddock

Production
Franz Wimmer
King Laurin GmbH
39057 St. Michael/Eppan, Italy

Printing
Fry Communications, Inc.
800 West Church Rd.
Mechanicsburg, PA 17055

Advertising Sales
Tammy Ferguson
832-717-4445 ext 26
tammy@codemag.com

Circulation & Distribution
General Circulation: EPS Software Corp.
Newsstand: The NEWS Group (TNG)
Media Solutions

Subscriptions
Subscription Manager
Colleen Cade
ccade@codemag.com

US subscriptions are US $29.99 for one year. Subscriptions outside the US are US $49.99. Payments should be made in US dollars drawn on a US bank. American Express, MasterCard, Visa, and Discover credit cards accepted. Bill me option is available only for US subscriptions. Back issues are available. For subscription information, e-mail subscriptions@codemag.com.

Subscribe online at
www.codemag.com

CODE Developer Magazine
6605 Cypresswood Drive, Ste 425, Spring, Texas 77379
Phone: 832-717-4445

On First Principles

Two of my favorite technical books are Andy Hunt's and Dave Thomas' book *The Pragmatic Programmer* and Bob Martin's *Clean Code: A Handbook of Agile Software Craftsmanship*. These books are borne of what I call *Actionable Theory*. We know that person, the "enterprise architect" type who graces us with their presence to offer up a cookbook commentary. I say "commentary" as opposed to "answer" because all too often, the observation is devoid of actionable context. And all too often, the sentences begin with "you should...." Once they provide their "advice," like a seagull, they fly away, only to return at some later time to repeat the cycle.

For the record, the above referenced books were published in 1999 and 2008 respectively, they've stuck around and are at least as much relevant today as they were decades ago. Recently, I've given much thought to theory and its role in software development. What we do is supposed to be utilitarian. Businesses use our work product. In general, people want to hear less about theory and more about "just getting the job done."

Imagine building a house, brick by brick, without regard to theory. What would we end up with? It might all work out. That, however, is more about faith and hope. Faith and hope are not strategies. Consider the case of the first Tacoma Narrows bridge which opened in July 1940. It collapsed in December 1940. There were plans and engineering principles applied. But a lot of shortcuts were taken in building the bridge. Even during construction, when winds picked up, the workers could feel the movement. That is why the bridge was nicknamed the "Galloping Gertie." Once things got bad enough, there was no issue with problem recognition. There was an issue with problem resolution. Long story short, five days after a course of action was decided upon, the bridge collapsed. The real issue was not the bridge's sensitivity to aerostatic flutter. The real issue was not adhering to first principles.

What principles do you apply to building software? Here are mine:

1. Theory Matters

Whether it's building a bridge, building, airplane, or software, those who put practice before theory do so at their own peril. We have a very recent example with the Boeing 737 Max 8. That's especially relevant because that tragedy represents a concrete example of the intersection between a physically engineered thing and software. There's much more to learn, but so far, it seems that not much was done correctly on that project. And like the Tacoma Narrows Bridge, it appears that there's an aggressive attempt at after-the-fact remedial steps. Can that work? Sure, anything is possible. Is it likely to work? Absent good tests, there is no quantifiable way to know whether something is more likely than not to work.

In software development, history has taught us that there are certain things, which, if applied, tend to coincide with success. Can there still be failure? Of course, there can be failure because other variables matter. For example, my code may be exceptionally beautifully written and does what it's supposed to do on my development rig. But if it's put on crap hardware in production, it won't matter much. Therefore, we test, whether it be unit-, load-, or performance-based.

Design patterns are another example of theory. Another good book to consider is Christopher Alexander's *A Pattern Language: Towns, Buildings, and Construction*. If you've ever used a wiki, you can thank Ward Cunningham for that. Ward's inspiration for the wiki was born out of the Portland Pattern Repository, which was a practical result of a 1987 OOPSLA paper titled: *Using Pattern Languages for Object-Oriented Programs*, and which drew heavily from the 1977 *A Pattern Language* book. Ward Cunningham collaborated with his wife Karen, a mathematician and Kent Beck. For the record, Ward and Kent are part of the crew that created the Agile Manifesto and are cited heavily in the *Clean Code* book.

Although we shouldn't sacrifice delivery for purity, at the same time, we shouldn't throw things over the wall for the sake of speed, just to get something delivered. If you're thinking of the classic project management troika of Good, Fast, and Cheap: you can only choose two—you get the point. We don't build bridges to withstand 500-year floods, as that isn't practical. But we do build bridges to withstand floods. Software should be no different.

Theory matters. And while there are many reasons software can fail, without exception, one of the root causes will almost always be a failure to heed theoretical principles.

2. People, Process, and Tools—In That Order

Ultimately, people must build things. People are the ones to develop and govern processes and to develop and employ tools. Two of my favorite words to describe what's necessary for good process are **Discipline** and **Rigor.** What we do is difficult. There must be discipline to "stick to it." At the same time, we must be flexible to change when needed. That's where rigor comes in. We should be as through as practicably possible—but no more so. I often go back to the Agile Manifesto and the associated principles that are often misinterpreted as meaning "no documentation." Nothing could be further from the truth. We simply value working software over *comprehensive* documentation. In other words, we strive to avoid the "bike-shedding" problem.

With people that can employ a good process, tools, whether they are developed or used, stand a chance. On tools, without good people and process, we end up with the "leaky abstraction." Where software is supposed to buffer you from certain details, it can't quite do so completely. The result is a tool that straddles the line and thus gets in the way and isn't as useful. You know these tools. The same holds true for throwing tools at a problem. Without good people and process, A), you don't know what the right tools are, and B), even if you did, the chances that you can rigorously evaluate and use them are low.

People and process can make up for poor tooling. Good tooling is wasted on poor people and process.

3. Do the Right Thing, in the Right Way, for the Right Reason

If you're going to cut corners, why are you doing it? Can you achieve the same end? If not, you may have to scale back so that you can do it the right way. This is just another flavor of the good-fast-cheap trilogy. It's just a different way of articulating that constraint where you can only pick two of the three items.

(Continued on page 79)

PLURALSIGHT

Upskill from anywhere

Build in-demand tech skills no matter where you are with access to courses on top technologies, skill assessments, paths and more.

Expert-led courses

Keep up with the pace of change with thousands of expert-led, in-depth courses.

Pluralsight IQ

Validate skill levels with assessments that take 10 minutes or less.

Guided learning

Measure progress towards personal objectives and learning goals.

Start for free today

pluralsight.com/codemag