THE MICROSOFT JOURNAL FOR DEVELOPERS

SEPTEMBER 2017 VOL 32 NO 9

msdn magazine

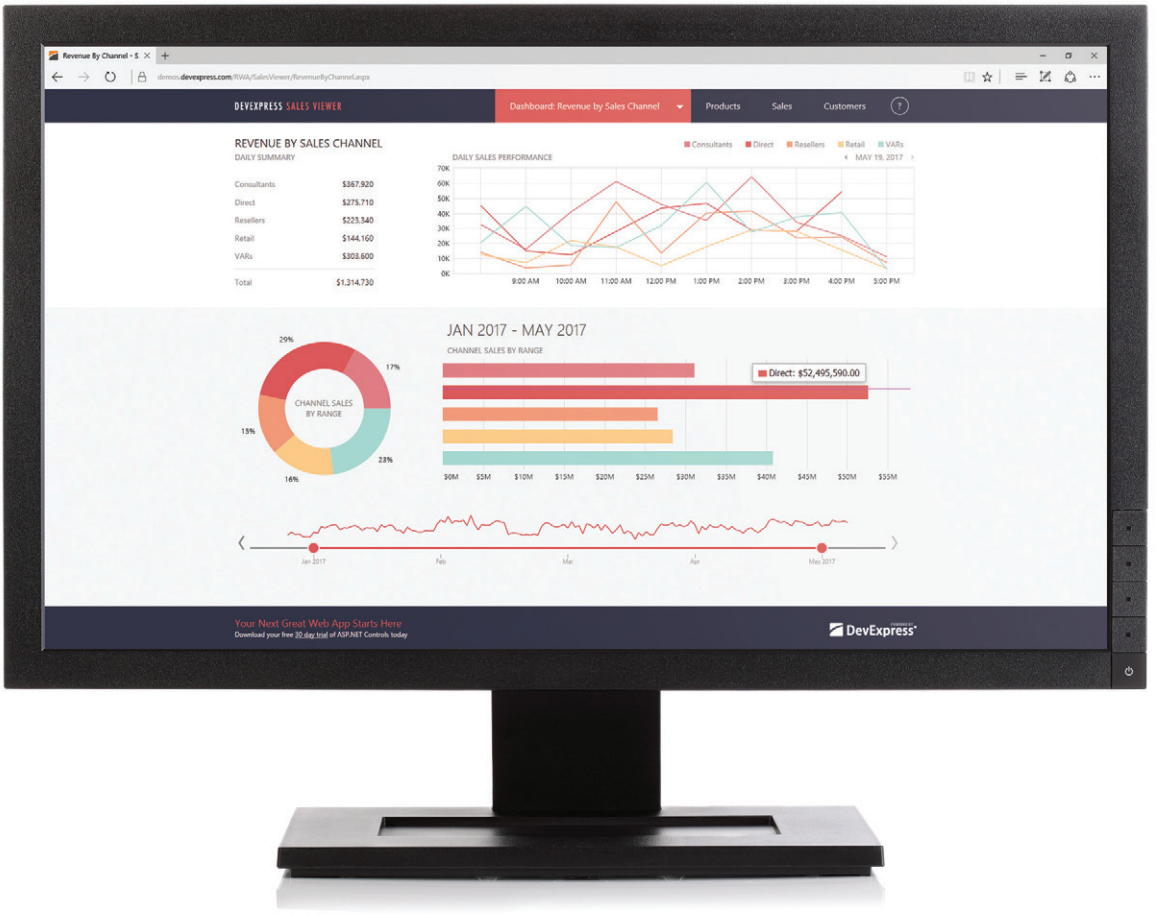
.NET Core

.NET Standard, .NET Core and ASP.NET Core...16, 22, 28

DevExpress®

Your Next Great Web App Starts Here

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



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

devexpress.com/try

THE MICROSOFT JOURNAL FOR DEVELOPERS

SEPTEMBER 2017 VOL 32 NO 9

msdn magazine

.NET Standard, .NET Core and ASP.NET Core...16, 22, 28

COLUMNS

Microsoft


Till 3am
Roger Deep
MUSIC BY GENRE
Hip Hop 25%
Country 35%
Rock & Roll 18%
Metal 25%
EDM 25%
Jazz 25%
FINANCE
Example Corporation (EXMPL)
23.18
2/15/2017, 9:55 AM
-0.34 (-0.72%)
OCT
SEP
SEP
SEP
Close
Contains...
22.34
22.288
21.9729
22.12
21.7834
1 - 5 of 10 records
Date
Contains...
Thu Jan 19 2017
Wed
Tue
Mon
Sun
High
Mail App
File
Active
Tab Item
Item Hover
Cut
Copy
Paste
Delete
Clipboard
Text Style
Find and Replace
Editing
inbox@mail.com
Inbox 13
Clutter
Drafts 1
Sent Items
Junk Email
From
Name LastName
Sed ut perspiciatis unde omnis iste natus
This is th
Sometimes it co
Very short
But if it's longer and it doesn't
This is the subject of the email
This is the subject of the email, it could be p
Reply
Reply All
This is the subject of the email
JULY 29
WEEK 31
Expense
100
0
S
M
T
W
Th
F
S
AVG. $45
TODAY
Coffee
@ STARBUCKS
$19.25
Movie
RISE OF GORT
4 75
YESTERDAY
Travel
4 75

msdn magazine

SEPTEMBER 2017 VOLUME 32 NUMBER 9

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

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

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

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

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

Reaching the Staff

Staff may be reached via e-mail, telephone, fax, or mail.
E-mail: To e-mail any member of the staff, please use the following form: FirstinitialLastname@1105media.com
Irvine Office (weekdays, 9:00 a.m. – 5:00 p.m. PT)
Telephone 949-265-1520; Fax 949-265-1528
4 Venture, Suite 150, Irvine, CA 92618
Corporate Office (weekdays, 8:30 a.m. – 5:30 p.m. PT)
Telephone 818-814-5200; Fax 818-734-1522
9201 Oakdale Avenue, Suite 101, Chatsworth, CA 91311
The opinions expressed within the articles and other contentsherein do not necessarily express those of the publisher.

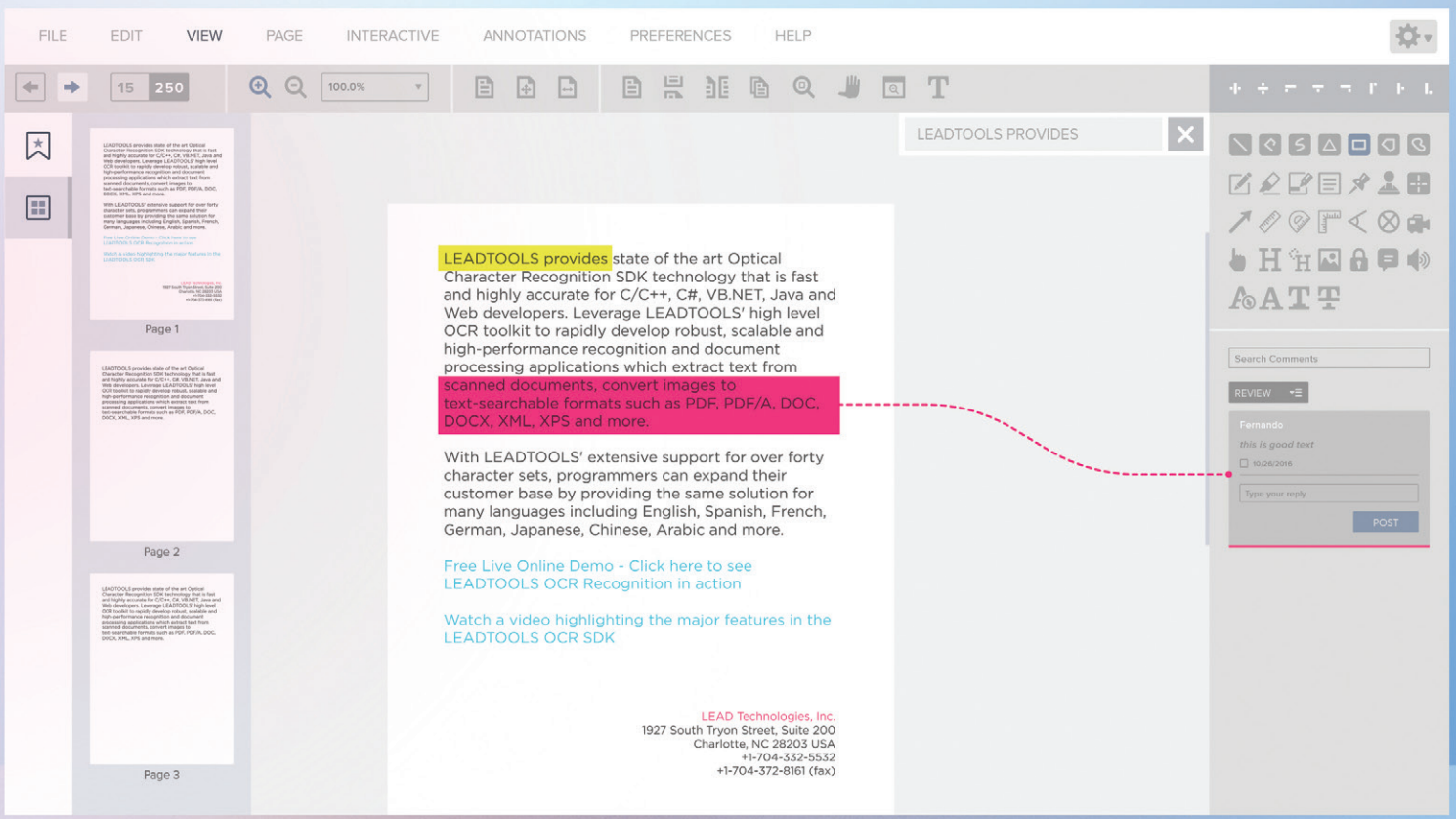
Microsoft

BPA WORLDWIDE
1906 2006 100 AMERICAN BUSINESS MEDIA
WPA Western Publications Association

LEADTOOLS®

THE WORLD LEADER IN IMAGING SDKs

The leading **Document Viewer SDK** just got better.



With only a few lines of code, developers can use the **LEADTOOLS** HTML5/JavaScript Document Viewer SDK to add rich document viewing features to any project, including text search, annotation, memory-efficient paging, inertial scrolling, and vector display.

- CREATE, ORGANIZE & EXPORT A DOCUMENT FROM MULTIPLE FILES
- PDF, DOC, DOCX, RTF, HTML, SVG, and XPS
- ANNOTATIONS & MARKUP
- DRAG AND DROP INTERFACE
- SEARCHABLE TEXT

SDKs for WEB | SDKs for MOBILE | SDKs for DESKTOP | SDKs for SERVER

.NET Windows API Java WinRT Linux iOS macOS Android JavaScript

DOWNLOAD OUR 60 DAY EVALUATION
WWW.LEADTOOLS.COM

LEAD TECHNOLOGIES INCORPORATED

SALES@LEADTOOLS.COM
800.637.1840

MICHAEL DESMOND

Hard Core: Focus on .NET Core 2.0 and .NET Standard 2.0

Microsoft has been pushing hard with its cross-platform, open source vision of software development, promoting the .NET Core and ASP.NET Core frameworks, and presenting .NET Standard as a universal target for .NET-based development. In August, Microsoft released version 2.0 of .NET Core, ASP.NET Core and .NET Standard, enabling a more robust, inclusive and complete platform for development than ever before.

To help support that effort, *MSDN Magazine* is publishing a set of five feature articles focused on the new releases. Immo Landwerth kicks off the festivities with his feature, "Demystifying .NET Core and .NET Standard," where he addresses some of the confusion about the differences between .NET Framework, .NET Core and .NET Standard, and shows how Microsoft has developed a strategic vision for coding across the various flavors of .NET. Andrew Hall and Joe Morris follow that up with a hands-on dive into writing .NET Core apps on Windows, macOS and Linux using tools like Visual Studio, Visual Studio for Mac and Visual Studio Code.

There's a lot going on in this issue and it reflects the excitement that Microsoft sees among developers in the field.

Web development gets a lot of love in this issue. Mike Rousos introduces the new ASP.NET Core 2.0 framework, guiding readers through development of a simple ASP.NET Core Web site as he explains important ASP.NET Core concepts and technologies. Steve Smith follows with "Simpler ASP.NET MVC Apps with Razor Pages," which shows how this new feature of ASP.NET Core reduces friction common to MVC development while preserving flexibility. Finally, Microsoft Program Manager Nikhil Joglekar's "Snapshot Debugging for Production Apps and Services in Azure" illustrates how developers can diagnose production code bugs using the new Snapshot Debugger with Visual Studio and App Insights.

There's a lot going on in this issue and it reflects the excitement that Microsoft sees among developers in the field. Andrew Hall says that .NET Core is matching the quickened pace of modern development, by decoupling framework updates from the Windows OS and providing the ability to deploy side-by-side and even per-application copies of the framework.

"We commonly hear of customers who are stuck on much older versions of the .NET Framework, because a few applications on the server don't work correctly with newer versions, which means that all their applications are pinned to the lowest common denominator," Hall says. ".NET Core removes the problem of a single machine-wide installation of the framework, freeing developers to use the latest features for their new apps, without fear of impacting older apps running on the server."

Microsoft has made a concerted effort to increase the utility and clarity of its framework offerings. Steve Smith notes that both .NET Core 2.0 and ASP.NET Core 2.0 eliminate rough edges in the cross-platform savvy frameworks and strongly boost productivity.

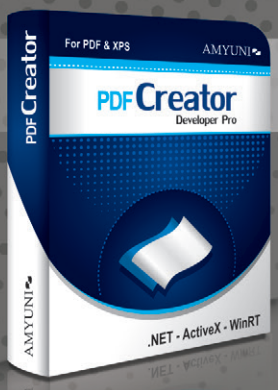
"There is more of an emphasis on convention over configuration, especially in areas like Startup, so that developers don't have to deal with quite as much plumbing code in the basic project templates. All of the functionality is still available, but the expected path is now set up by default with much less code," says Smith.

Sitting atop all this is the .NET Standard specification of APIs, which Immo Landwerth says brings consistency to .NET-based development across .NET Framework, .NET Core and Xamarin. Developers who target .NET Standard 2.0 can build libraries that can be used from any .NET implementation or supported OS, including Windows, Linux, Android, macOS and iOS. If you want to maximize the reach of your code in the most consistent and reliable way, .NET Standard is the way to do it.

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

Switch to Amyuni PDF

AMYUNI



Create and Edit PDFs in .NET, COM/ActiveX, WinRT & UWP

NEW v5.5

- Edit, process and print PDF 1.7 documents
- Create, fill-out and annotate PDF forms
- Fast and lightweight 32- and 64-bit components for .NET and ActiveX/COM
- New Universal Apps DLLs enable publishing C#, C++, CX or Javascript apps to windows Store
- Updated Postscript/EPS to PDF conversion module

Complete Suite of Accurate PDF Components

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

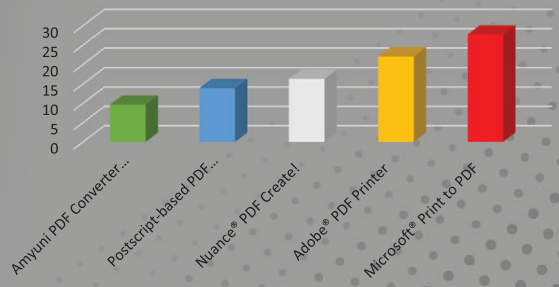
Other Developer Components from Amyuni®

- WebkitPDF: Direct conversion of HTML files into PDF and XAML without the use of a web browser or a printer driver
- PDF2HTML5: Conversion of PDF to HTML5 including dynamic forms
- Postscript to PDF Library: For document workflow applications that require processing of Postscript documents
- OCR Module: Free add-on to PDF Creator uses the Tesseract engine for character recognition
- Javascript engine: Integrate a full Javascript interpreter into your applications to process PDF files or for any other need

High Performance PDF Printer for Desktops and Servers

- Print to PDF in a fraction of the time needed with other tools. WHQL tested for all Windows platforms. Version 5.5 updated for Windows 10 support

Benchmark Testing - Amyuni vs Others
Seconds required to convert a document to PDF



CERTIFIED FOR Windows Server® 2008

Windows Server 2012 Certified

Windows 7

Windows

All development tools available at

www.amyuni.com

AMYUNI Technologies

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

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

DINO ESPOSITO

Cookies, Claims and Authentication in ASP.NET Core

Most of the literature concerning the theme of authentication in ASP.NET Core focuses on the use of the ASP.NET Identity framework. In that context, things don't seem to have changed much or, more precisely, all the changes that occurred in the infrastructure have been buried in the folds of the framework so that it looks nearly the same on the surface.

If you look at user authentication in ASP.NET Core outside the comfortable territory of ASP.NET Identity, you might find it quite different from what it was in past versions. ASP.NET Identity is a full-fledged, comprehensive, big framework that's overkill if all you need is to authenticate users via plain credentials from a simple database table. In this case, you'll see that the overall approach to authentication is still based on familiar concepts such as principal, login form, challenge and authorization attributes, except that the way you implement them is radically different. In this month's column, I'll explore the cookie authentication API as made available in ASP.NET Core, including the core facts of external authentication.

Foundation of ASP.NET Authentication

In ASP.NET, user authentication involves the use of cookies. Any users that attempt to visit a private page are redirected to a login page if they don't carry a valid authentication cookie. The login page, after having verified provided credentials, emits the cookie, which then travels with any subsequent requests from that user through the same browser until it expires. This is the same basic workflow you might know from past versions of ASP.NET. In ASP.NET Core, it only looks different because of the different middleware and the different configuration of the runtime environment.

Figure 1 **Registering Middleware for Cookie Authentication**

```
// This code is for ASP.NET Core 1.x
public void Configure(IApplicationBuilder app)
{
  app.UseCookieAuthentication(new CookieAuthenticationOptions
  {
    AutomaticAuthenticate = true,
    AutomaticChallenge = true,
    AuthenticationScheme = "Cookies",
    CookieName = "YourAppCookieName",
    LoginPath = new PathString("/Account/Login"),
    ExpireTimeSpan = TimeSpan.FromMinutes(60),
    SlidingExpiration = true,
    ReturnUrlParameter = "original",
    AccessDeniedPath = new PathString("/Account/Denied")
  });
}
```

There are two major changes in ASP.NET Core for those coming from an ASP.NET Web Forms and ASP.NET MVC background. First, there's no longer a web.config file, meaning that configuration of the login path, cookie name and expiration is retrieved differently. Second, the IPrincipal object—the object used to model user identity—is now based on claims rather than the plain user name. To enable cookie authentication in a brand-new ASP.NET Core 1.x application, you first reference the Microsoft.AspNetCore.Authentication.Cookies package and then add the code snippet in **Figure 1**.

Most of the information that classic ASP.NET MVC applications stored in the <authentication> section of the web.config file are configured as middleware options. The snippet in **Figure 1** comprehends canonical options you might want to choose. **Figure 2** explains each in more detail.

Note that path properties are not of type string. LoginPath and AccessDeniedPath are of type PathString, which, compared to the plain String type, provides correct escaping when building a request URL.

The overall design of the user authentication workflow in ASP.NET Core gives you an unprecedented amount of flexibility. Every aspect of it can be customized at will. As an example, let's see how you can control the authentication workflow being used on a per-request basis.

Dealing with Multiple Authentication Schemes

By setting AutomaticChallenge to false, you instruct the middleware not to react to the [Authorize] challenges by performing, say, a redirect. If no middleware will handle the challenge an exception is thrown. Automatic challenge was the norm in past versions of ASP.NET and there was almost nothing you could do about it.

In ASP.NET Core, you can register multiple and distinct pieces of authentication middleware and determine either algorithmically or via configuration which middleware has to be used for each request. When multiple authentication middleware is used, AutomaticAuthenticate can be true on multiple middleware. AutomaticChallenge, instead, should only be enabled on zero or one middleware. For more details, see bit.ly/2tS07Sm.

Common examples of authentication middleware are cookie-based authentication, bearer authentication, authentication through social networks, or an identity server and whatever else you can ever think to implement. Suppose you have the following code in the Configure method of your startup class:

```
app.UseCookieAuthentication(new CookieAuthenticationOptions()
{
  AuthenticationScheme = "Cookies",
  LoginPath = new PathString("/Account/Login/"),
  AutomaticAuthenticate = true
});
app.UseIdentityServerAuthentication(new IdentityServerAuthenticationOptions
{
  AuthenticationScheme = "Bearer",
  AutomaticAuthenticate = true
}
```

Be aware that there are constants to be used in place of magic strings like "Cookies" just to limit typos. In particular, the string "Cookies" can be replaced as below:

```
AuthenticationScheme = CookieAuthenticationDefaults.AuthenticationScheme
```

Note that UseIdentityServerAuthentication isn't part of the ASP.NET Core framework but belongs to the Identity Server framework (see github.com/IdentityServer). To choose the authentication scheme on a per-request basis, you use a new attribute on the Authorize attribute that in ASP.NET MVC marks actions as subject to authentication and authorization:

```
[Authorize(ActiveAuthenticationSchemes = "Bearer")]
public class ApiController : Controller
{
  // Your API action methods here
  ...
}
```

The net effect of the code snippet is that all public endpoints of the sample ApiController class are subject to the identity of the user as authenticated by the bearer token.

Modeling the User Identity

In ASP.NET, the IPrincipal interface defines the software contract that defines the core of the user identity. The logged user is exposed through the User property of the HttpContext controller property. IPrincipal has the same implementation in ASP.NET 4.x (including ASP.NET MVC) and ASP.NET Core. However, in ASP.NET Core the default principal object isn't GenericPrincipal, but the new ClaimsPrincipal type. The difference is relevant.

GenericPrincipal wraps up one key piece of user information—the user name—even though custom user data can be added to the authentication ticket encrypted in the cookie. Over the years, the sole user name has become too little for the needs of modern applications. The role of the user, as well as some other chunks of information, most noticeably picture and display name, appear absolutely required today, forcing every realistic application to create its own custom principal type or query user information for each and every request using the user name as the key. ClaimsPrincipal just brilliantly solves the problem.

A claim is a key/value pair that describes a property of the logged user. The list of properties is up to the application, but includes name and role at the very minimum. Claims, in other words, are the ASP.NET Core way to model identity information. Claims can be read from any source, whether databases, cloud or local storage, even hardcoded. The Claim class is as simple as this:

```
public class Claim
{
  public string Type { get; }
  public string Value { get; }

  // More properties ...
}
```

The login process in ASP.NET Core passes through three classes: Claim, ClaimIdentity and ClaimsPrincipal.

Collecting the list of claims is a simple matter of populating an array of Claim objects:

```
public Claims[] LoadClaims(User user)
{
  var claims = new[]
  {
    new Claim(ClaimTypes.Name, user.UserName),
    new Claim(ClaimTypes.Role, user.Role),
    new Claim("Picture", user.Picture)
  };
  return claims;
}
```

The name of the claim is a plain descriptive name rendered as a string. However, most common claim types have been grouped as constants into the ClaimTypes class. Before you create the principal, which is required to call the authentication workflow completed, you must get hold of an identity object:

```
var identity = new ClaimsIdentity(claims, "Password");
```

The first argument is self-explanatory—the list of claims associated with the identity being created. The second argument is a string that refers to the authentication scheme required to verify the identity. The string "Password" is a reminder of what will be required by the system for a user to prove her identity. The string "Password" is informational only and not a syntax element.

Figure 2 **Cookie Authentication Options**

Property	Description
AccessDeniedPath	Indicates the path where an authenticated user will be redirected if the provided identity doesn't have permission to view the requested resource. The same as getting an HTTP 403 status code.
AutomaticAuthenticate	Indicates the middleware runs on every request and attempts to validate cookie and build an identity object from content.
AutomaticChallenge	Indicates the middleware redirects the browser to a login page if the user isn't authenticated or to the access denied page if the user is authenticated but not authorized on the requested resource.
AuthenticationScheme	Name of the middleware. This property works in conjunction with AutomaticChallenge to selectively pick up the authentication middleware on a per-request basis.
CookieName	Name of the authentication cookie being created.
ExpireTimeSpan	Sets the expiration time of the authentication cookie. Whether the time has to be intended as absolute or relative is determined by the value of the SlidingExpiration property.
LoginPath	Indicates the path where an anonymous user will be redirected to sign in with her own credentials.
ReturnUrlParameter	Name of the parameter being used to pass the originally requested URL that caused the redirect to the login page in case of anonymous users.
SlidingExpiration	Indicates whether the ExpireTimeSpan value is absolute or relative. In the latter case, the value is considered as an interval and the middleware will reissue the cookie if more than half the interval has elapsed.

Another interesting aspect of the previous example is that an explicit user name is not strictly required. Any claim, regardless of the declared type, can be used to name the user. The following code snippet shows another equivalent way to have a new identity object:

```
var identity = new ClaimsIdentity(claims,
  CookieAuthenticationDefaults.AuthenticationScheme,
  "Nickname",
  ClaimTypes.Role);
```

The new identity has "Cookies" as the authentication scheme and Nickname is the name of the claim in the provided list to be used to provide the name of the user. Role, instead, is the name of the claim in the same list determining the role. If not specified, the last two parameters default to ClaimTypes.Name and ClaimTypes.Role. Finally, you create the principal from the identity. It's worth noting, though, that a principal may have multiple identities. If it sounds weird, think that different areas of the same application might need different information to authenticate the user in much the same way an ID is required to identify yourself at some hotel desks and an electronic key to get into the elevator. The ClaimsPrincipal class has both an Identities property (a collection) and an Identity property. The latter is only a reference to the first item in the collection.

External Authentication

ASP.NET Core supports external authentication via identity providers from the ground up. Most of the time, all you do is install the appropriate NuGet package for the task. To rely on Twitter for authenticating users, you bring in the Microsoft.AspNetCore.Authentication.Twitter package and install the related middleware:

```
app.UseTwitterAuthentication(new TwitterOptions()
{
  AuthenticationScheme = "Twitter",
  SignInScheme = "Cookies",
  ConsumerKey = "...",
  ConsumerSecret = "..."
});
```

The SignInScheme property is the identifier of the authentication middleware that will be used to persist the resulting identity. In the example, an authentication cookie will be used. To see its effects, you first add a controller method to call into Twitter:

```
public async Task TwitterAuth()
{
  var props = new AuthenticationProperties
  {
    RedirectUri = "/"
  };
  await HttpContext.Authentication.ChallengeAsync("Twitter", props);
}
```

Next, once Twitter has successfully authenticated the user, the SignInScheme property instructs the application on what to do next. A value of "Cookies" is acceptable if you want a cookie out of the claims returned by the external provider (Twitter, in the example). If you want to review and complete the information through, say, an intermediate form, then you have to break the process in two, introducing a temporary sign-in scheme. In addition to the standard cookie middleware you have the following:

```
app.UseCookieAuthentication(new CookieAuthenticationOptions
{
  AuthenticationScheme = "ExternalCookie",
  AutomaticAuthenticate = false
});
app.UseTwitterAuthentication(new TwitterOptions
{
  AuthenticationScheme = "Twitter",
  SignInScheme = "ExternalCookie"
});
```

When the external provider returns, a temporary cookie is created using the ExternalCookie scheme. Having set the redirect path appropriately, you have a chance to inspect the principal returned by Twitter and edit it further:

```
var props = new AuthenticationProperties
  RedirectUri = "/account/external"
};
```

To complete the workflow you also need to sign in in the cookies scheme and sign out of temporary scheme (ExternalCookie):

```
public async Task<IActionResult> External()
  var principal = await HttpContext
    .Authentication
    .AuthenticateAsync("ExternalCookie");
  // Edit the principal
  ...

  await HttpContext.Authentication.SignInAsync("Cookies", principal);
  await HttpContext.Authentication.SignOutAsync("ExternalCookie ");
}
```

ExternalCookie, as well as cookies, are just internal identifiers and can be renamed as long as they remain consistent throughout the application.

Wrapping Up

In ASP.NET Core many things seem to be radically different, but in the end most of the concepts you might know from ASP.NET remain unchanged. You still need to have an authentication cookie created, and you can still control the name of the cookie and the expiration date. External authentication is supported and login pages have the same structure as before. However, configuration and underlying working of the authentication infrastructure are different, while retaining their previous flexibility.

Everything stated in this article refers to ASP.NET Core 1.x. There are a few things that will work differently in ASP.NET Core 2.0. In particular, authentication middleware is now exposed as services and must be configured on ConfigureServices:

```
services.AddCookieAuthentication(options =>
{
  Options.LoginPath = new PathString("/Account/Login"),
    options.AutomaticAuthenticate = true,
    options.AutomaticChallenge = true,
    options.AuthenticationScheme = "Cookies",
    ...
});
```

In the Configure method of the Startup class of ASP.NET Core 2.0 applications, you just declare your intention to use authentication services without any further options:

```
app.UseAuthentication();
```

Also note that the SignInAsync method you use in your code to create the authentication cookie is also exposed from the HttpContext object directly, instead of passing through an intermediate Authentication property as shown in the last code snippet for ASP.NET Core 1.x. ■

Dino Esposito *is the author of "Microsoft .NET: Architecting Applications for the Enterprise" (Microsoft Press, 2014) and "Modern Web Applications with ASP.NET" (Microsoft Press, 2016). A technical evangelist for the .NET and Android platforms at JetBrains, and frequent speaker at industry events worldwide, Esposito shares his vision of software at software2cents@wordpress.com and on Twitter: @despos.*

Thanks to the following Microsoft technical expert for reviewing this article: *Chris Ross*

JULIE LERMAN

DDD-Friendlier EF Core 2.0

If you've been following this column for a while, you may have noticed quite a few articles about implementing Entity Framework (EF) when building solutions that lean on Domain-Driven Design (DDD) patterns and guidance. Even though DDD is focused on the domain, not on how data is persisted, at some point you need data to flow in and out of your software.

In past iterations of EF, its patterns (conventional or customized) have allowed users to map uncomplicated domain classes directly to the database without too much friction. And my guidance has generally been that if the EF mapping layer takes care of getting your nicely designed domain models into and out of the database without having to create an additional data model, this is sufficient. But at the point when you find yourself tweaking your domain classes and logic to make them work better with Entity Framework, that's a red flag that it's time to create a model for data persistence and then map from the domain model classes to the data model classes.

I was a little surprised to realize how long ago those articles about DDD and EF mappings came out. It's been four years since I wrote the three-part series called "Coding for Domain-Driven Design: Tips for Data-Focused Devs," which span the August, September and October 2013 issues of *MSDN Magazine*. Here's a link to the first part, which includes links to the entire series: msdn.com/magazine/dn342868.

There were two specific columns that addressed DDD patterns and explained how EF did or didn't easily map between your domain classes and database. As of EF6, one of the biggest issues was the fact that you couldn't encapsulate and thereby protect a "child" collection. Using the known patterns for protecting the collection (most often this meant employing an IEnumerable) didn't align with EF's requirements, and EF wouldn't even recognize that the navigation should be part of the model. Steve Smith and I spent a lot of time thinking about this when we created our Pluralsight course, Domain-Driven Design Fundamentals (bit.ly/PS-DDD) and eventually Steve came up with a nice workaround (bit.ly/2ufw89D).

EF Core finally solved this problem with version 1.1 and I wrote about this new feature in the January 2017 column (msdn.com/magazine/mt745093). EF Core 1.0 and 1.1 also resolved a few other DDD constraints but left some gaps—most notably the inability to map DDD value objects used in your domain types. The ability to do this had existed in EF since its beginning, but it hadn't yet been brought to EF Core. But with the upcoming EF Core 2.0, that limitation is now gone.

Code download available at bit.ly/2tDRXwi.

What I'm going to do in this article is lay out the EF Core 2.0 features available to you that align with many of the DDD concepts. EF Core 2.0 is much friendlier to developers who are leveraging these concepts and perhaps it will introduce you to them for the first time. Even if you don't plan to embrace DDD, you can still benefit from its many great patterns! And now you can do that even more with EF Core.

One-to-One Gets Smarter

In his book, "Domain-Driven Design," Eric Evans says, "A bidirectional association means that both objects can be understood only together. When application requirements do not call for traversal in both directions, adding a traversal direction reduces interdependence and simplifies the design." Following this guidance has indeed removed side effects in my code. EF has always been able to handle uni-directional relationships with one-to-many and one-to-one. In fact, while writing this article, I learned that my longtime misunderstanding that a one-to-one relationship with both ends required forces you into a bi-directional relationship was wrong. However, you did have to explicitly configure those required relationships, and that's something you don't have to do now in EF Core, except for with edge cases.

> EF Core should be able to correctly infer the dependent end of the relationship based on the foreign key property.

An unfavorable requirement in EF6 for one-to-one relationships was that the key property in the dependent type had to double as the foreign key back to the principal entity. This forced you to design classes in an odd way, even if you got used to it. Thanks to the introduction of support for unique foreign keys in EF Core, you can now have an explicit foreign key property in the dependent end of the one-to-one relationship. Having an explicit foreign key is more natural. And in most cases, EF Core should be able to correctly infer the dependent end of the relationship based on the existence of that foreign key property. If it doesn't get it right

DevExpress®

DevExpress Spreadsheet for WPF & WinForms

with Chart, Pivot Table and Cell Editor support

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

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

Free 30-day trial
devexpress.com/spreadsheet

#UseTheBest

because of some edge case, you'll have to add a configuration, which I'll demonstrate shortly when I rename the foreign key property.

To demonstrate a one-to-one relationship, I'll use my favorite EF Core domain: classes from the movie "Seven Samurai":

```
public class Samurai {
  public int Id { get; set; }
  public string Name { get; set; }
  public Entrance Entrance { get; set; }
}

public class Entrance {
  public int Id { get; set; }
  public string SceneName { get; set; }
  public int SamuraiId { get; set; }
}
```

Now with EF Core, this pair of classes—Samurai and Entrance (the character's first appearance in the movie)—will be correctly identified as a uni-directional one-to-one relationship, with Entrance being the dependent type. I don't need to include a navigation property in the Entrance and I don't need any special mapping in the Fluent API. The foreign key (SamuraiId) follows convention, so EF Core is able to recognize the relationship.

EF Core infers that in the database, Entrance.SamuraiId is a unique foreign key pointing back to Samurai. Keep in mind something I struggled with because (as I have to continually remind myself), EF Core is not EF6! By default, .NET and EF Core will treat the Samurai.Entrance as an optional property at run time unless you have domain logic in place to enforce that Entrance is required. Starting with EF4.3, you had the benefit of the validation API that would respond to a [Required] annotation in the class or mapping. But there is no validation API (yet?) in EF Core to watch for that particular problem. And there are other requirements that are database-related. For example, Entrance.SamuraiId will be a non-nullable int. If you try to insert an Entrance without a SamuraiId value populated, EF Core won't catch the invalid data, which also means that the InMemory provider currently doesn't complain. But your relational database should throw an error for the constraint conflict.

From a DDD perspective, however, this isn't really a problem because you shouldn't be relying on the persistence layer to point out errors in your domain logic. If the Samurai requires an Entrance, that's a business rule. If you can't have orphaned Entrances, that's also a business rule. So the validation should be part of your domain logic anyway.

For those edge cases I suggested earlier, here's an example. If the foreign key in the dependent entity (for example, Entrance) doesn't follow convention, you can use the Fluent API to let EF Core know. If Entrance.SamuraiId was, perhaps Entrance.SamuraiFK, you can clarify that FK via:

```
modelBuilder.Entity<Samurai>().HasOne(s=>s.Entrance)
  .WithOne().HasForeignKey<Entrance>(e=>e.SamuraiFK);
```

If the relationship is required on both ends (that is, Entrance must have a Samurai) you can add IsRequired after WithOne.

Properties Can Be Further Encapsulated

DDD guides you to build aggregates (object graphs), where the aggregate root (the primary object in the graph) is in control of all of the other objects in the graph. That means writing code that prevents other code from misusing or even abusing the rules. Encapsulating properties so they can't be randomly set (and, often, randomly read) is a key method of protecting a graph. In EF6 and earlier, it was always possible to make scalar and navigation properties have private setters and still be recognized by EF when it read and updated data, but you couldn't easily make the properties private. A post by Rowan Miller shows one way to do it in EF6 and links back to some earlier workarounds (bit.ly/2eHTm2t). And there was no true way to protect a navigation collection in a one-to-many relationship. Much has been written about this latter problem. Now, not only can you easily have EF Core work with private properties that have backing fields (or inferred backing fields), but you can also truly encapsulate collection properties, thanks to support for mapping IEnumerable<T>. I wrote about the backing fields and IEnumerable<T> in my previously mentioned January 2017 column, so I won't rehash the details here. However, this is very important to DDD patterns and therefore relevant to note in this article.

While you can hide scalars and collections, there's one other type of property you may very well want to encapsulate—navigation properties. Navigation collections benefit from the IEnumerable<T> support, but navigation properties that are private, such as Samurai.Entrance, can't be comprehended by the model. However, there is a way to configure the model to comprehend a navigation property that's hidden in the aggregate root.

For example, in the following code I declared Entrance as a private property of Samurai (and I'm not even using an explicit backing field, though I could if needed). You can create a new Entrance with the CreateEntrance method (which calls a factory method in Entrance) and you can only read an Entrance's SceneName property. Note that I'm employing the C# 6 null-conditional operator to prevent an exception if I haven't yet loaded the Entrance:

```
private Entrance Entrance {get;set;}
public void CreateEntrance (string sceneName) {
    Entrance = Entrance.Create (sceneName);
  }
public string EntranceScene => Entrance?.SceneName;
```

By convention, EF Core won't make a presumption about this private property. Even if I had the backing field, the private Entrance wouldn't be automatically discovered and you wouldn't be able to use it when interacting with the data store. This is an intentional API design to help protect you from potential side effects. But you can configure it explicitly. Remember that when Entrance is public, EF Core is able to comprehend the one-to-one relationship. However, because it's private you first need to be sure that EF knows about this.

Figure 1 **The PersonName Value Object**

```
public class PersonName : ValueObject<PersonName> {
  public static PersonName Create (string first, string last) {
    return new PersonName (first, last);
  }
  private PersonName () { }
  private PersonName (string first, string last) {
    First = first;
    Last = last;
  }
  public string First { get; private set; }
  public string Last { get; private set; }
  public string FullName => First + " " + Last;
}
```

Data Quality Made Easy.
Your Data, Your Way.

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

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

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

Microsoft® SQL Server® Integration Services

Microsoft Dynamics® CRM

Microsoft® .NET

Microsoft Azure

Microsoft® Excel

Start Your Free Trial

www.Melissa.com/msft-pd

Melissa Data is Now Melissa.
Why the change?
See for Yourself at the New www.Melissa.com

melissa™

1-800-MELISSA

In OnModelCreating, you need to add the HasOne/WithOne fluent mapping to make EF Core aware. Because Entrance is private, you can't use a lambda expression as a parameter of HasOne. Instead, you have to describe the property by its type and its name. WithOne normally takes a lambda expression to specify the navigation property back to the other end of the pairing. But Entrance doesn't have a Samurai navigation property, just the foreign key. That's fine! You can leave the parameter empty because EF Core now has enough information to figure it out:

```
modelBuilder.Entity<Samurai> ()
  .HasOne (typeof (Entrance), "Entrance").WithOne();
```

What if you use a backing property, such as _entrance in the Samurai class, as shown in these changes:

```
private Entrance _entrance;
private Entrance Entrance { get{return _entrance;} }
public void CreateEntrance (string sceneName) {
    _entrance = _entrance.Create (sceneName);
  }
public string EntranceScene => _entrance?.SceneName;
```

EF Core will figure out that it needs to use the backing field when materializing the Entrance property. This is because, as Arthur Vickers explained in the very long conversation we had on GitHub while I was learning about this, if "there is a backing field and there is no setter, EF just uses the backing field [because] there is nothing else it can use." So it just works.

If that backing field name doesn't follow convention, if, for example, you named it _foo, you will need a metadata configuration:

```
modelBuilder.Entity<Samurai> ()
  .Metadata
  .FindNavigation ("Entrance")
  .SetField("_foo");
```

Now updates to the database and queries will be able to work out that relationship. Keep in mind that if you want to use eager loading, you'll need to use a string for Entrance becaise it can't be discovered by the lambda expression; for example:

```
var samurai = context.Samurais.Include("Entrance").FirstOrDefault();
```

You can use the standard syntax for interacting with backing fields for things like filters, as shown at the bottom of the Backing Fields documentation page at bit.ly/2vgDwPk.

Value Objects Are Now Supported

Value objects are an important concept for DDD as they allow you to define domain models as value types. A value object doesn't have its own identity and becomes part of the entity that uses it as a property. Consider the string value type, which is made up of a series of characters. Because changing even a single character changes the meaning of the word, strings are immutable. In order to change a string, you must replace the entire string object. DDD guides you to consider using value objects anywhere you've identified a one-to-one relationship. You can learn more about value objects in the DDD Fundamentals course I mentioned earlier.

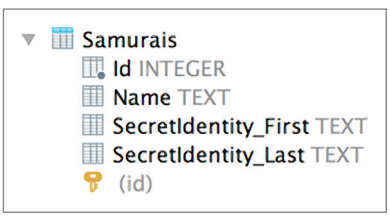


Figure 2 **The Schema of the Samurais Table, Including the Properties of the Value Object**

EF always supported the ability to include value objects through its ComplexType type. You could define a type with no key and use that type as a property of an entity. That was enough to trigger EF to recognize it as a ComplexType and map its properties into the table to which the entity is mapped. You could then extend the type to also have features required of a value object, such as ensuring the type is immutable and a means to assess every property when determining equality and overriding the Hash. I often derive my types from Jimmy Bogard's ValueObject base class to quickly adopt these attributes.

A person's name is a type that's commonly used as a value object. You can ensure that any time someone wants to have a person's name in an entity, they always follow a common set of rules. **Figure 1** shows a simple PersonName class that has First and Last properties—both fully encapsulated—as well as a property to return a FullName. The class is designed to ensure both parts of the name are always supplied.

> Until EF Core 2.0, there was no feature similar to complex types, so you couldn't easily use them without adding in a separate data model.

I can use PersonName as a property in other types and continue to flesh out additional logic in the PersonName class. The beauty of the value object over a one-to-one relationship here is that I don't have to maintain the relationship when I'm coding. This is standard object-oriented programming. It's just another property. In the Samurai class, I've add a new property of this type, made its setter private and provided another method named Identify to use instead of the setter:

```
public PersonName SecretIdentity{get;private set;}
public void Identify (string first, string last) {
  SecretIdentity = PersonName.Create (first, last);
}
```

Until EF Core 2.0, there was no feature similar to ComplexTypes, so you couldn't easily use value objects without adding in a separate data model. Rather than just reimplement the ComplexType in EF Core, the EF team created a concept called owned entities, which leverages another EF Core feature, shadow properties. Now, owned entities are recognized as additional properties of the types that own them and EF Core understands how they resolve in the database schema and how to build queries and updates that respect that data.

EF Core 2.0 convention won't automatically discover that this new SecretIdentity property is a type to be incorporated into the persisted data. You'll need to explicitly tell the DbContext that the Samurai.SecretIdentity property is an owned entity in DbContext.OnModelCreating using the OwnsOne method:

```
protected override void OnModelCreating (ModelBuilder modelBuilder) {
  modelBuilder.Entity<Samurai>().OwnsOne(s => s.SecretIdentity);
}
```

This forces the properties of PersonName to resolve as properties of Samurai. While your code will use the Samurai.SecretIdentity type and navigate through that to the First and Last properties, those two properties will resolve as columns in the Samurais database table. EF Core convention will name them with the name of the property in Samurai (SecretIdentity) and the name of the owned entity property, as shown in **Figure 2**.

Now I can identify a Samurai's secret name and save it with code similar to this:

```
using (var context = new SamuraiContext()) {
  var samurai = new Samurai { Name = "HubbieSan"
  samurai.Identify ("Late", "Todinner");
  context.Samurais.Add (samurai);
  context.SaveChanges ();
}
```

In the data store, "Late" gets persisted into the SecretIdentity_First field and "Todinner" into the SecretIdentity_Last field.

Then I can simply query for a Samurai:

```
var samurai=context.Samurais .FirstOrDefaultAsync (s => s.Name == "HubbieSan")
```

EF Core will assure that the resulting Samurai's SecretIdentity property is populated and I can then see the identity by requesting:

```
samurai.SecretIdentity.FullName
```

EF Core requires that properties that are owned entities are populated. In the sample download, you'll see how I designed the PersonName type to accommodate that.

Simple Classes for Simple Lessons

What I've shown you here are simple classes that leverage some of the core concepts of a DDD implementation in the most minimal way that lets you see how EF Core responds to those constructs. You've seen that EF Core 2.0 is able to comprehend one-to-one uni-directional relationships. It can persist data from entities where scalar, navigation and collection properties are fully encapsulated. It also allows you to use value objects in your domain model and is able to persist those, as well.

For this article, I've kept the classes simple and lacking in additional logic that more correctly constrains the entities and value objects using DDD patterns. This simplicity is reflected in the download sample, which is also on GitHub at bit.ly/2tDRXwi. There you'll find both the simple version and an advanced branch where I've tightened down this domain model and applied some additional DDD practices to the aggregate root (Samurai), its related entity (Entrance) and the value object (PersonName) so you can see how EF Core 2.0 handles a more realistic expression of a DDD aggregate. In an upcoming column, I'll discuss the advanced patterns applied in that branch.

Please keep in mind that I'm using a version of EF Core 2.0 shortly before its final release. While most of the behaviors I've laid out are solid, there's still the possibility of minor tweaks before 2.0.0 is released. ■

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

Thanks to the following Microsoft technical expert for reviewing this article: *Arthur Vickers*

Demystifying .NET Core and .NET Standard

Immo Landwerth

As the newest members of the .NET family, there's much confusion about .NET Core and .NET Standard and how they differ from the .NET Framework. In this article, I'll explain exactly what each of these are and look at when you should choose each one.

Before going into detail, it's helpful to look at the larger picture of .NET to see where .NET Core and .NET Standard fit in. When.NET Framework first shipped 15 years ago, it had a single .NET stack that you could use for building Windows desktop and Web applications. Since then, other .NET implementations have emerged, such as Xamarin, which you can use for building mobile apps for iOS and Android, as well as macOS desktop applications, as shown in **Figure 1**.

Here's how .NET Core and .NET Standard fit into this:

- **.NET Core:** This is the latest .NET implementation. It's open source and available for multiple OSes. With .NET Core, you can build cross-platform console apps and ASP.NET Core Web applications and cloud services.
- **.NET Standard:** This is the set of fundamental APIs (commonly referred to as base class library or BCL) that all .NET implementations must implement. By targeting .NET Standard, you can build libraries that you can share across all your .NET apps, no matter on which .NET implementation or OS they run.

This article discusses:

- Using the "dotnet" command-line tool to create, build and test .NET Core projects
- Using .NET Standard to build libraries that run on any .NET implementation
- Reusing .NET Framework libraries through the .NET Standard 2.0 compatibility mode

Technologies discussed:

.NET Standard, .NET Core, .NET CLI

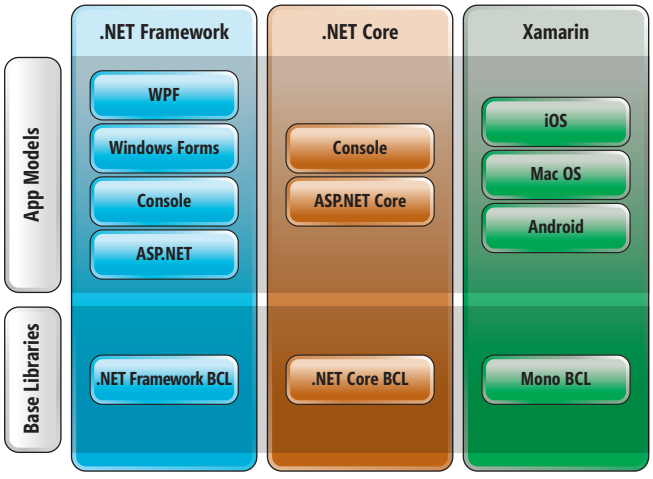


Figure 1 **The .NET Landscape**

Introduction to .NET Core

.NET Core is a new cross-platform and fully open source .NET implementation that was forked from .NET Framework and Silverlight. It's optimized for mobile and server workloads by enabling self-contained XCOPY deployments.

To get a better feel for .NET Core, let's take a closer look at what developing for .NET Core looks like. And let's do this while exploring the new command line-based tooling. You can also use Visual Studio 2017 for your .NET Core development, but because you're reading this article, chances are you're quite familiar with Visual Studio already, so I'll focus on the new experience.

When .NET was created, it was heavily optimized for rapid application development on Windows. In practice, this means that .NET development and Visual Studio were inseparable friends. And sure thing: Using Visual Studio for development is a blast. It's super productive and the debugger is hands down the best I've ever used.

However, there are cases where using Visual Studio isn't the most convenient option. Let's say you want to just play with .NET to learn C#: You shouldn't have to download and install a multi-gigabyte IDE. Or, say you're accessing a Linux machine over SSH where using an IDE is simply not an option. Or, say you're simply someone who prefers using a command-line interface (CLI).

That's why a first-class CLI was created, called .NET Core CLI. The .NET Core CLI's main driver is called "dotnet." You can use it for virtually all aspects of your development, including creating, building, testing and packaging projects. Let's see what this looks like.

Start by creating and running a Hello World console app (I use PowerShell on Windows, but this will work equally well with Bash on macOS or Linux):

```
$ dotnet new console -o hello
$ cd hello
$ dotnet run
Hello World!
```

The "dotnet new" command is the CLI equivalent of File | New Project in Visual Studio. You can create a variety of different project types. Type "dotnet new" to see the different templates that come pre-installed.

Now, let's extract some of the logic into a class library. To do this, first create a class library project that's parallel to your hello project:

```
$ cd ..
$ dotnet new library -o logic
$ cd logic
```

The logic you want to encapsulate is the construction of a Hello World message, so change the contents of Class1.cs to the following code:

```
namespace logic
{
  public static class HelloWorld
  {
      public static string GetMessage(string name) => $"Hello {name}!";
  }
}
```

At this point, you should also rename Class1.cs to HelloWorld.cs:

```
$ mv Class1.cs HelloWorld.cs
```

Note that you don't have to update the project file for this change. The new project files used in .NET Core simply include all source files from the project's directory. Thus, adding, removing and renaming files doesn't require modifying the project anymore. This makes file operations smoother from the command line.

Figure 2 **Updating the Program.cs File to Use HelloWorld Class**

```
using System;
using logic;

namespace hello
{
  class Program
  {
    static void Main(string[] args)
    {
      Console.Write("What's your name: ");
      var name = Console.ReadLine();
      var message = HelloWorld.GetMessage(name);
      Console.WriteLine(message);
    }
  }
}
```

Figure 3 **Changing the UnitTest1.cs Contents to Add a Test**

```
using System;
using Xunit;
using logic;

namespace tests
{
  public class UnitTest1
  {
    [Fact]
    public void Test1()
    {
      var expectedMessage = "Hello Immo!";
      var actualMessage = HelloWorld.GetMessage("Immo");
      Assert.Equal(expectedMessage, actualMessage);
    }
  }
}
```

To use the HelloWorld class, you need to update the hello app to reference the logic library. You can do this by editing the project file or by using the dotnet add reference command:

```
$ cd ../hello
$ dotnet add reference ../logic/logic.csproj
```

Now, update the Program.cs file to use the HelloWorld class, as shown in **Figure 2**.

To build and run your app, just type dotnet run:

```
$ dotnet run
What's your name: Immo
Hello Immo!
```

You can also create tests from the command line. The CLI supports MSTest, as well as the popular xUnit framework. Let's use xUnit in this example:

```
$ cd ..
$ dotnet new xunit -o tests
$ cd tests
$ dotnet add reference ../logic/logic.csproj
```

Change the UnitTest1.cs contents, as shown in **Figure 3**, to add a test.

Now you can run the tests by invoking dotnet test:

```
$ dotnet test
Total tests: 1. Passed: 1. Failed: 0. Skipped: 0.
Test Run Successful.
```

To make things a bit more interesting, let's create a simple ASP.NET Core Web site:

```
$ cd ..
$ dotnet new web -o web
$ cd web
$ dotnet add reference ../logic/logic.csproj
```

Edit the Startup.cs file and change the invocation of app.Run to use the HelloWorld class as follows:

Figure 4 **The Project Structure You Created**

```
$ tree /f
|
├──hello
|       hello.csproj
|       Program.cs
|
├──logic
|       HelloWorld.cs
|       logic.csproj
|
├──tests
|       tests.csproj
|       UnitTest1.cs
|
└──web
        Program.cs
        Startup.cs
        web.csproj
```

```
app.Run(async (context) =>
{
  var name = Environment.UserName;
  var message = logic.HelloWorld.GetMessage(name);
  await context.Response.WriteAsync(message);
});
```

To start the development Web server, just use dotnet run again:

```
$ dotnet run
Hosting environment: Production
Now listening on: http://localhost:5000
Application started. Press Ctrl+C to shut down.
```

Browse to the displayed URL, which should be http://localhost:5000.

At this point, your project structure should look like **Figure 4**.

To make it easier to edit the files using Visual Studio, let's also create a solution file and add all the projects to the solution:

```
$ cd ..
$ dotnet new sln -n HelloWorld
$ ls -fi *.csproj -rec | % { dotnet sln add $_.FullName }
```

As you can see, the .NET Core CLI is powerful and results in a lean experience that developers from other backgrounds should find quite familiar. And while you used dotnet with PowerShell on Windows, the experience would look quite similar if you were on Linux or macOS.

Another huge benefit of .NET Core is that it supports self-contained deployments. You could containerize your application using Docker in such a way that it has its own copy of the .NET Core runtime. This lets you run different applications on the same machine using different .NET Core versions without them interfering with each other. Because .NET Core is open source, you can also include nightly builds or even versions you've modified or built yourself, potentially including modifications you made. However, that's beyond the scope of this article.

Introduction to .NET Standard

When you're building modern experiences, your app often spans multiple form factors and, therefore, multiple .NET implementations. In this day and age, customers pretty much expect that they can use your Web app from their mobile phone and that data can be shared via a cloud-based back end. When using a laptop, they also want to get access via a Web site. And for your own infrastructure, you likely want to use command-line tools and potentially even desktop apps for letting your staff manage the system. See **Figure 5** for how the different .NET implementations play into this.

In such an environment, code sharing becomes a major challenge. You need to understand where APIs are available and make sure that shared components only use APIs that are available across all .NET implementations you're using.

And that's where .NET Standard comes in. .NET Standard is a specification. Each .NET Standard version defines the set of APIs that all .NET implementations must provide to conform to that version. You can think of it as yet-another .NET stack, except that you can't build apps for it, only libraries. It's the .NET implementation you should use for libraries that you want to reference from everywhere.

You're probably wondering which APIs .NET Standard covers. If you're familiar with .NET Framework, then you should be familiar with the BCL, which I mentioned earlier. The BCL is the set of fundamental APIs that are independent of UI frameworks and application models. It includes the primitive types, file I/O, networking, reflection, serialization, XML and more.

All .NET stacks implement some version of .NET Standard. The rule of thumb is that when a new version of a .NET implementation is produced, it will usually implement the latest available version of the .NET Standard.

A good analogy is HTML and browsers: Think of the HTML specification as the .NET Standard and the different browsers as the .NET implementations, such as .NET Framework, .NET Core and Xamarin.

At this point, you're probably curious how you can use .NET Standard. In fact, you already have. Remember when we created the logic class library earlier? Let's take a closer look at the project file:

```
$ cd logic
$ cat logic.csproj
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <TargetFramework>netstandard2.0</TargetFramework>
  </PropertyGroup>

</Project>
```

Let's contrast this with the "hello" console application project file:

```
$ cd ..\hello
$ cat hello.csproj
<Project Sdk="Microsoft.NET.Sdk">

  <ItemGroup>
    <ProjectReference Include="..\logic\logic.csproj" />
  </ItemGroup>

  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>netcoreapp2.0</TargetFramework>
  </PropertyGroup>

</Project>
```

Figure 5 **Descriptions of .NET Implementations**

	OS	Open Source	Purpose
.NET Framework	Windows	No	Used for building Windows desktop applications and ASP.NET Web apps running on IIS.
.NET Core	Windows, Linux, macOS	Yes	Used for building cross-platform console apps and ASP.NET Core Web apps and cloud services.
Xamarin	iOS, Android, macOS	Yes	Used for building mobile applications for iOS and Android, as well as desktop apps for macOS.
.NET Standard	N/A	Yes	Used for building libraries that can be referenced from all .NET implementations, such as .NET Framework, .NET Core and Xamarin.

As you can see, the logic library has a value for TargetFramework of netstandard2.0, while the console app has a value of netcoreapp2.0. The TargetFramework property indicates which .NET implementation you're targeting. So, the console app is targeting .NET Core 2.0, while the library is targeting .NET Standard 2.0. That means you can reference the logic library not only from a .NET Core app, but also from an app built for .NET Framework or Xamarin.

Unfortunately, most of the libraries available today aren't targeting .NET Standard, yet. Most of them are targeting the .NET Framework. Of course, not all libraries can (or even should) target .NET Standard. For instance, a library containing Windows Presentation Foundation (WPF) controls needs to target .NET Framework because UI isn't part of the standard. However, many of the general-purpose libraries only target .NET Framework because they were created when .NET Standard simply didn't exist yet.

With .NET Standard 2.0, the API set is large enough so that most, if not all, of the general-purpose libraries can target .NET Standard. As a result, 70 percent of all the libraries that exist on NuGet today only use APIs that are now part of .NET Standard. Still, only a fraction of them are explicitly marked as being compatible with .NET Standard.

To unblock developers from using them, a compatibility mode has been added. If you install a NuGet package that doesn't offer a library for your target framework, nor provides one for .NET Standard, NuGet will try to fall back to .NET Framework. In other words, you can reference .NET Framework libraries as if they were targeting .NET Standard.

I'll show you what this looks like. In my example, I'll use a popular collection library called PowerCollections, which was written in 2007. It wasn't updated in a while and still targets .NET Framework 2.0. I'll install this from NuGet into the hello app:

```
$ dotnet add package Huitian.PowerCollections
```

This library provides additional collection types that the BCL doesn't provide, such as a bag, which makes no ordering guarantees. Let's change the hello app to make use of it, as shown in **Figure 6**.

If you run the program, you'll see the following:

```
$ dotnet run
hello.csproj : warning NU1701: Package 'Huitian.PowerCollections 1.0.0' was
restored using '.NETFramework,Version=v4.6.1' instead of the project target
framework '.NETCoreApp,Version=v2.0'. This may cause compatibility problems.
1
3
2
```

So, what just happened? The hello app is targeting .NET Core 2.0. Because .NET Core 2.0 implements .NET Standard 2.0, it also has the compatibility mode for referencing .NET Framework libraries. However, not all .NET Framework libraries will work on all .NET implementations. For example, they might use Windows Forms or WPF APIs. NuGet has no way of knowing that, so it gives you a warning message so you're aware of this situation and don't waste your time troubleshooting issues that might result from this.

Figure 6 **Sample Application Using PowerCollections**

```
using System;
using Wintellect.PowerCollections;

namespace hello
{
  class Program
  {
    static void Main(string[] args)
    {
      var data = new Bag<int>() { 1, 2, 3 };
      foreach (var element in data)
        Console.WriteLine(element);
    }
  }
}
```

Note that you'll get this warning each time you build. This avoids the problem where you simply didn't see the warning during package installation, or forgot about it.

Of course, there's nothing worse than unactionable warnings that you need to overlook every time you build. So, the idea here is that after you validate your app, you can then disable the warning for that package. Because the app is running fine (it correctly printed the contents of the bag you created) you can now suppress the warning. To do that, edit the hello.csproj file and add the NoWarn attribute to the package reference:

```
<PackageReference Include="Huitian.PowerCollections" Version="1.0.0"
  NoWarn="NU1701" />
```

If you now run the app again, the warning should be gone. Should you install another package that uses the compatibility mode, you'll get the warning for that package for which you can suppress, as well.

> The new tooling also lets class library projects produce NuGet packages as part of the build.

The new tooling also lets class library projects produce NuGet packages as part of the build. This makes it much simpler to share your libraries with the world (by pushing to nuget.org) or just within your organization (by pushing to your own package feed on Visual Studio Team Services or MyGet). The new projects also support multi-targeting, which lets you build a single project for multiple .NET implementations. This means you can use conditional compilation (#if) to adapt the library to specific .NET implementations. It also lets you build .NET Standard wrappers for platform-specific APIs. However, all of that is beyond the scope of this article.

Wrapping Up

.NET Standard is a specification of APIs that all .NET implementations must provide. It brings consistency to the .NET family and enables you to build libraries you can use from any .NET implementation. It replaces PCLs for building shared components.

.NET Core is an implementation of the .NET Standard that's optimized for building console applications, Web apps and cloud services using ASP.NET Core. Its SDK comes with a powerful tooling that in addition to Visual Studio development supports a full command line-based development workflow. You can learn more about them at aka.ms/netstandardfaq and aka.ms/netcore. ■

Immo Landwerth *is a program manager at Microsoft, working on .NET. He focuses on .NET Standard, the BCL and API design.*

Thanks to the following Microsoft technical experts for reviewing this article: *Phillip Carter, Richard Lander and Maira Wenzel*

File Format APIs

Working with Files?

CREATE CONVERT PRINT

MODIFY COMBINE

FREE TRIAL

Aspose.Total

Manipulate Word, Excel, PDF, PowerPoint, Outlook and more than 100 other file formats in your applications without installing Microsoft Office.

DOC, XLS, PDF, PPT, MSG, BMP, PNG, XML and many more!

Platforms supported: .NET, Java, Cloud, Android, SharePoint, Reporting Services, and JasperReports

ASPOSE
File Format APIs

CONTACT US

US: +1 903 306 1676
EU: +44 141 628 8900
AU: +61 2 8006 6987

sales@asposeptyltd.com

Try for FREE at
www.aspose.com

Write .NET Apps How and Where You Want

Andrew Hall and Joe Morris

Hopefully you've heard by now that the Microsoft .NET isn't just for Windows anymore. .NET Core brings the ability to write applications that run on any OS you choose (Windows, macOS or Linux) using the language you prefer (C#, Visual Basic or F#). Of course, with the ability to write applications targeting the platform of your choice, you expect the same great developer tools that have always been a hallmark of .NET development. The great news is, as our supported platforms have grown, so have the tools available for creating great .NET applications.

In this article, we'll walk you through the development tools available based on your personal work style and OS. We'll start with the command-line tools included with the SDK that you can pair with any editor (although we recommend Visual Studio Code), then we'll show the great new capabilities of Visual Studio 2017 before introducing you to our newest member of the Visual Studio family, Visual Studio for Mac. Finally, we'll conclude by showing how these tools empower you to leverage software containers and try your apps seamlessly in the Microsoft Azure cloud.

Basic Concepts

For the purposes of this article, we'll use the term ".NET Core" everywhere for the sake of consistency, but there are two important technologies covered by this name: .NET Core and .NET Standard. .NET Core is the cross-platform runtime that executes applications. .NET Standard is a compiler-enforced set of APIs that are used as a target for class libraries, so a single library can run on any runtime that supports the standard. For example, .NET Core 2.0, .NET Framework 4.6.1 and Mono 4.8 support .NET Standard 2.0 (and previous) class libraries. For a better understanding of this concept, see the companion article, "Demystifying .NET Core and .NET Standard" in this issue.

It's also worth noting the languages supported by .NET Core and .NET Standard 2.0. At a high level, both technologies support C#, Visual Basic and F# in 2.0. However, our aim with Visual Basic this release was to enable .NET Standard 2.0 class libraries. This means Visual Basic only offers templates for class libraries and console apps, while C# and F# also include templates for ASP.NET Core 2.0 apps.

This article discusses:

- The simplified common project format
- The command-line interface
- Unit testing support
- Visual Studio Code, Visual Studio, Visual Studio for Mac
- Using software containers and the Microsoft Azure cloud

Technologies discussed:

.NET Core 2.0, Visual Studio 2017, Visual Studio Code, Visual Studio for Mac, Docker, Microsoft Azure

Improved Common Project File

One of the first things you'll notice about working with .NET Core projects is that they share a highly simplified common project format. This means that a project created with one tool (for example, the command-line tools) will work on anything that supports .NET Core (such as Visual Studio or Visual Studio for Mac). With our new project file format, we've removed the need for GUIDs, and for explicitly listing included files—which greatly reduces merge conflicts when committing updates to version control. The following code shows the entire contents of the project file for a new C# .NET Core console app:

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>netcoreapp2.0</TargetFramework>
  </PropertyGroup>
</Project>
```

Additionally, the common format means that members of the same team can work using the device and platform of their choice. If one team member wants to work on a Mac, another on Windows, and a third on Linux, they can all seamlessly contribute to the same project. To learn more about the updated project format, visit aka.ms/newprojectfile.

Command-Line Tools

The foundation of any developer tool is the Software Development Kit (SDK), and .NET Core is no different. One of the integral parts of the .NET Core SDK is the command-line interface (CLI), which enables you to create, build and run apps from the command line.

With the CLI you can use your favorite editor to create apps with no need to install heavier-weight tools. To get started, install the free .NET Core SDK (available from dot.net).

Once the SDK is installed, open your favorite command prompt and run "dotnet --help" to see a list of all the commands available to you. The dotnet command is the driver for the .NET CLI, where the grammar is "dotnet <verb> [parameters]". For example, to see all the project type templates available to create, type "dotnet new." To create a new project, provide the short name as the parameter to the "new" action; for example, "dotnet new razor" will create a C# ASP.NET Core Web App with the new .NET Core 2.0 Razor Pages (to learn about this great new feature, see aka.ms/razorpages). To run the application, type "dotnet run" and it will build and start the application. Opening a browser to the URL indicated in the output string (for example, http://localhost:5000) lets you interact with the app you're building. From this point, development is a simple matter of choosing your favorite editor to build and run the app.

Unit Testing

Once you have an app, you'll want to add unit testing support relatively early in the lifecycle. Unit testing works with your IDEs as you'd expect, but you can also build and run unit tests directly from the .NET CLI.

You may have noticed that "dotnet new" offers two different unit-testing project types: xunit and mstest. To get started, create a unit test project with your preferred unit test testing framework, for example "dotnet new xunit," and then in your test project folder type "dotnet add reference <path to project to test>" to have the CLI add the reference to the project file for you. Once you have a unit-testing project, run your test projects from the CLI using the "dotnet test" command. Regardless of which project type you chose, both xunit and mstest projects are supported by Visual Studio and Visual Studio for Mac, as well as the CLI.

Visual Studio Code

While you have the freedom to use any editor you wish, we think Visual Studio Code offers the best experience for users who desire a lightweight editing experience in conjunction with the .NET Core CLI. Visual Studio Code is our lightweight cross-platform editor with source control and debugging support built in. Using the .NET CLI with Visual Studio Code, you can build .NET applications on any platform you choose. To get started you'll need to install the .NET Core SDK as previously discussed. Once you have it on your machine, install Visual Studio Code from code.visualstudio.com, and then install the C# extension from Microsoft—C# for Visual Studio Code (powered by OmniSharp)—by selecting the extension tab on the left side of Visual Studio Code. (If you're using F#, you'll want to install the Ionide-fsharp extension, as well. Note that the C# extension is needed for .NET Core build and debugging support.) You're now ready to develop .NET Core projects using Visual Studio Code.

The C# extension enables Visual Studio code to call the CLI on your behalf to build and run applications for debugging. Once you've created your project, open Visual Studio Code and from the File menu choose Open Folder. You'll see a prompt when you open a C# file (.cs)—a suggestion bar at the top of the editor offering to generate the required assets for building and debugging, as shown in **Figure 1**. Choose Yes and it will generate two files, tasks.json, which enables building from Visual Studio Code, and launch.json, which enables debugging. You're now ready to edit, compile and debug directly from Visual Studio Code.

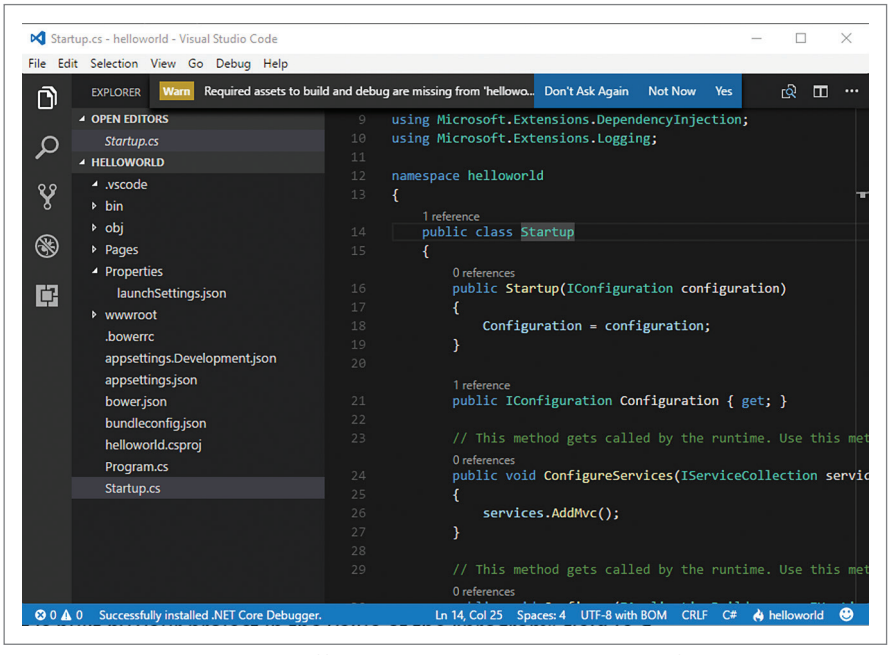


Figure 1 **Visual Studio Code Offering to Generate Required Files for Building and Debugging**

Visual Studio

Visual Studio continues to support a world-class developer experience, and the tooling support for .NET Core is no different. .NET Core 2.0 support was introduced in Visual Studio 2017 (version 15.3). To get started writing a .NET Core 2.0 app in Visual Studio, you need to download and install the .NET Core 2.0 SDK. Once it's on your machine, restart Visual Studio and you'll see .NET Core 2.0 as a target for new projects.

Our goal was to make Visual Studio 2017 the most productive development experience we've ever offered. To hit a few highlights, we've added code navigation improvements, a significant number of new refactoring/quick fix commands, code-style enforcement and Live Unit Testing.

Navigation and Refactoring We know that as you're working in your code it's important to have great support for navigating around your code base. The most popular improvements to code navigation in Visual Studio 2017 are:

- Go To Implementation (Ctrl+F12): Navigates from any base type or member to its various implementations.

- Go To All (Ctrl+T or Ctrl+,): Navigates directly to any file/type/member/symbol declaration. You can use the icons along the top of the feature to filter your result list or use the query syntax (for example, "f searchTerm" for files, "t searchTerm" for types and so on).
- Find All References (Shift+F12): Now with syntax colorization, Find All References results can be custom grouped by a combination of project, definition and path. You can also "lock" results so you can continue to find other references without losing your original results.
- Indent Guides: Dotted, gray, vertical lines act as landmarks in code to provide context within your frame of view. You may recognize these from the popular Productivity Power Tools.

Additionally, it's not enough to be able to simply navigate your code base; we know that when you find places in your code to change or clean up, you need tools to help you with that refactoring. To assist with this, we've added support for moving a type to a file with same name, syncing a file and type name, adding a null-check for a parameter, adding a parameter, adding a missing switch/Select case, making a method synchronous, converting a method to property (and vice versa), and resolving a merge conflict—just to name a few.

Code Style In Visual Studio 2017, you can configure and enforce your team's coding conventions to drive consistency across your entire repository with EditorConfig. EditorConfig is an open file format and we worked with its community to support .NET code style within this format. Teams can configure convention preferences and choose how they're enforced inside the editor (as suggestions, warnings or errors). The rules apply to whatever files are in the directory that contains the EditorConfig file. If you have different conventions for different projects, you can define each project's rules in different EditorConfig files as long as the projects are in separate directories. Because, at the end of the day, EditorConfig is just a text file, so it's easy to check it into source control and have it live and travel with your source. To learn more about EditorConfig support in Visual Studio, see aka.ms/editorconfig.

Live Unit Testing Once you have an MSTest or xUnit or NUnit test project, you can enable Live Unit Testing (LUT) for your .NET Core projects. LUT is a new feature introduced in Visual Studio 2017 Enterprise Edition. When you enable LUT, you get unit test coverage and pass/fail feedback, live in the code editor as you type, as shown in **Figure 2**. This means you no longer need to leave the editor to run unit tests to verify code changes. As soon as you type, you'll see instant feedback in the editor that shows the results for all tests that are impacted by the code change.

Teams that are passionate about quality and users who love test-driven development (TDD) will love this new addition. To turn on LUT, navigate to the Test entry in the Visual Studio menu bar, and select Live Unit Testing | Start.

Visual Studio for Mac

Visual Studio for Mac is the newest member of the Visual Studio family for developers who want an IDE experience on the macOS. Visual Studio for Mac has evolved from Xamarin Studio, which brings support for C# and F# projects. To get started, first install the .NET Core 2.0 SDK (available to download from dot.net), choose Create Project, then select the App entry under the .NET Core category to choose your desired project template.

You'll find you have a fully featured IDE for developing .NET Core apps on your Mac, as shown in **Figure 3**. One of the things we're working hard at is to bring a consistent experience between Visual Studio and Visual Studio for Mac. At this point, you have most of the features you've come to rely on in Visual Studio, including IntelliSense, code navigation, refactoring, unit testing and source control integration. For Web development, we've brought the Visual Studio HTML, CSS and JSON editors to Visual Studio for Mac. You'll notice that Razor files (.cshtml) and JavaScript/TypeScript are not currently supported, but we're working to bring that support to Visual Studio for Mac in a future update.

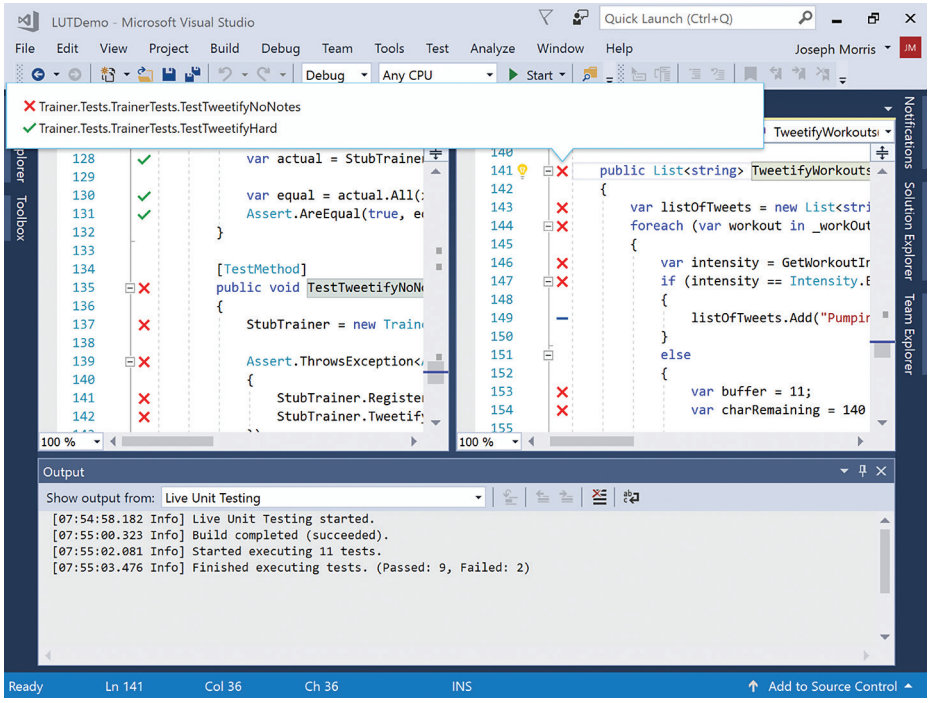


Figure 2 **Live Unit Testing Running on a .NET Core Project**

Tools with the Modern Cloud in Mind

As the world of technology evolves, one thing is clear—more and more apps are going to be designed to run in the cloud from the start. This means that your tools need to enable design patterns and practices with the modern cloud in mind, as well as enable you to quickly move an app from your local development machine to the cloud. With this goal in mind, both Visual Studio and Visual Studio for Mac have built-in support for publishing directly to Microsoft Azure, and for packaging applications as Docker containers.

msdn magazine VP

MSDN MAGAZINE VENDOR PROFILE

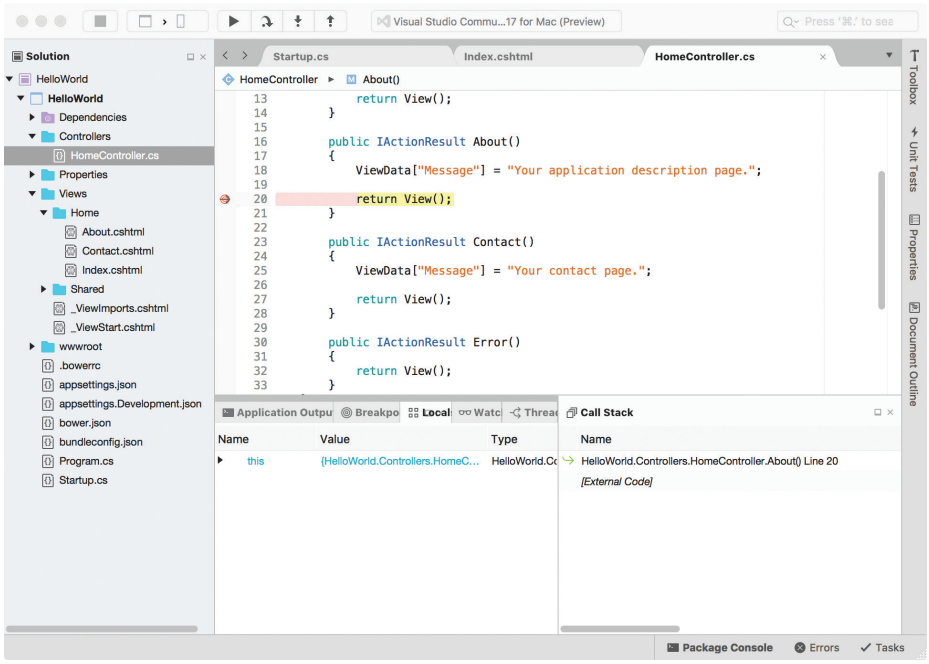


Figure 3 **Visual Studio for Mac**

Publish Directly to Azure When you're ready to move an application to run in the cloud, you'll likely want to begin by trying it in the cloud with as little effort as possible. Both Visual Studio and Visual Studio for Mac enable you to publish an app directly from your machine to Azure App Service. App Service is a fully managed cloud environment that enables you to run your app without the need to worry about complex configuration or infrastructure management.

Publishing your app from your local machine to Azure App Service from Visual Studio or Visual Studio for Mac is as simple as right-clicking the project and choosing Publish. You'll be asked to enter some information, such as the unique URL for your app, and you'll be able to choose either an existing App Service Plan (how many resources you want reserved for the collection of member apps) or create a new one if you don't have one yet. Once you've entered the information, your app will be running in the cloud in a matter of minutes.

Container Development Tools One of the things that's becoming clear about modern cloud development is that containers are going to revolutionize the way people architect and build software. Containers enable you to package your application and all of its dependencies, including a copy of the runtime, into a single unit, guaranteeing that changes to the underlying server will never break your application. This enables a microservice architecture pattern in which you deploy each unit of logic within your application as a separate container, with a defined protocol to communicate between them. This lets you scale high-demand parts of the app as needed without paying the cost of scaling the entire application, and it means that fixes affect only the container that's being updated rather than the entire application.

With that in mind, one of our goals with .NET Core is to make it the premier runtime for creating containerized microservices. Visual Studio, Visual Studio for Mac and Visual Studio Code all support building apps as Docker containers. (Note that at the time of this writing, Docker tooling support for Visual Studio for Mac requires the Docker Tools extension, and that Microsoft offers a Docker extension for Visual Studio Code to support creating Docker containers.)

To build a Docker container, you need Docker's tools for your OS installed (learn more and download from docker.com). Once you have the requirements installed, all you need to do in Visual Studio or Visual Studio for Mac is right-click the project, choose Add and then select Docker Support. This will add a docker-compose project to your solution, and a Dockerfile to the project. These assets enable your project to be built into a Docker container. When you then run the app from the IDE, you'll be running and debugging your application inside a container rather than on the host machine directly. The great news is that the publishing to Azure App Service experience we discussed also supports publishing Linux Docker containers.

Wrapping It Up

If you haven't tried building a .NET Core app yet, there's no better time to start than now. .NET Core and .NET Standard 2.0 (as covered in the "Demystifying .NET Core and .NET Standard" article) make creating a .NET application easy no matter what tools, OS, language or work style you choose.

We hope you found this tour of the available .NET Core tools useful. You can continue to follow our team's progress and future improvements on our blog at aka.ms/dotnetblog. Please try out the tools we discussed and give us feedback on what we can do to further improve your experience creating .NET Core apps. You can leave comments and questions on our blog posts, or send us feedback directly through Visual Studio and Visual Studio for Mac by using the Report a Problem and Provide a Suggestion features built into the IDEs. ■

Andrew Hall *is the program manager lead for the .NET languages and tools in Visual Studio. After graduating from college, he wrote line-of-business applications before returning to school for his master's degree in computer science. After completing his master's degree, he joined the diagnostics team in Visual Studio where he worked on the debugger, profiling and code analysis tools. He then moved to the .NET languages and tools team where he works on the .NET languages and supporting tools, including tools for ASP.NET, Web and Azure App Service. You can reach Hall via Twitter: @AndrewBrianHall.*

Joe Morris *is a program manager for .NET Core tools and Live Unit Testing in Visual Studio. He holds a master's degree in Computer Science. He started his career as an application developer consultant to enterprises in the United States in the 1990s. He joined Microsoft in 1997 with Microsoft Consulting Services, Iowa, and later moved to Redmond. In the last two years his focus has been on static code analysis and developer productivity tools. You can reach Morris via Twitter: @_jomorris.*

Thanks to the following Microsoft technical experts who reviewed this article: *Dustin Campbell, Phillip Carter, Livar Cunha, Mikayla Hutchinson, Mads Kristensen, Jordan Matthiesen and Kasey Uhlenhuth*

GrapeCity | Developer Solutions

Empowering Developers

GrapeCity.com

Achieve more with GrapeCity Developer Solutions.
Developers seek out GrapeCity and our great line of products because we get the fast-moving pace of their world. We live, breathe, and work within the developer's universe. Our community is your community.

We're your trusted partners, problem solvers, and team members.
Because we're developers, we understand the unique challenges you and your enterprise face. Our team and tools empower your development by decreasing your time, money, and security risk so you can spend more time developing innovative business solutions. GrapeCity's high-quality, fast, flexible UI controls and solutions have small footprints, key features, and universal APIs—all designed to reduce your learning curve and enhance your end product. And with direct access to our global product experts, you'll find that you're never alone in your goal to develop meaningful, intuitive applications for your enterprise.

ComponentOne
.NET UI CONTROLS

ActiveReports
REPORTING SOLUTIONS

Spread
SPREADSHEET SOLUTIONS

Wijmo
JAVASCRIPT UI CONTROLS

Getting Started with ASP.NET Core 2.0

Mike Rousos

ASP.NET Core makes it easy to create fast, portable, cross-platform Web applications. This article will guide you through developing a simple ASP.NET Core Web site and will show what role each file in the project plays. It will also explain important ASP.NET Core concepts along the way. There will be a special focus on changes in ASP.NET Core 2.0 to help readers familiar with ASP.NET Core 1.0 and 1.1 make the transition to 2.0.

Creating an ASP.NET Core Project

ASP.NET Core projects can be created from templates using either Visual Studio or the .NET Core command-line interface (.NET CLI). Visual Studio 2017 gives a great .NET Core development experience—with top-notch debugging, Docker integration and many other features—but I'll use the .NET CLI and Visual Studio Code in this walk-through, in case some of you want to follow along on a Mac or Linux dev machine.

The dotnet new command is used to create new .NET Core projects. Running dotnet new without any additional arguments will list the available project templates, as shown in **Figure 1**. If you're familiar with previous versions of the .NET CLI, you'll notice a number of new templates available in version 2.0.

Angular and React.js SPA Templates: These templates create an ASP.NET Core application that serves a single-page application (using either Angular 4 or React.js) as its front end. The templates include both the front-end and back-end applications, as well as a Webpack configuration to build the front end (and csproj modifications to kick off the Webpack build every time the ASP.NET Core project is built).

ASP.NET Core Web App with Razor Pages: Razor Pages is a new ASP.NET Core 2.0 feature that allows you to create pages that can handle requests directly (without needing a controller). These are a great option for scenarios that fit a page-based programming model.

For this walk-through, let's start with the new Razor Pages template by executing dotnet new razor. As soon as the project has been created, you should be able to run it by executing dotnet run. In previous versions of .NET Core, it would've been necessary to execute dotnet restore first to install the necessary NuGet packages. But beginning with .NET Core 2.0, the restore command is now automatically run by CLI commands that depend on it. Go ahead and test the template Web site out by executing dotnet run and navigating to the URL on which the app is listening (likely http://localhost:5000). You should see the Web app rendered (as in **Figure 2**).

Congratulations on launching your first ASP.NET Core 2.0 app! Now that you have a simple Web app running, let's take a

This article discusses:

- Creating an ASP.NET Core project
- Creating and running the Web host
- Dependency injection and the request-processing pipeline
- What's new in ASP.NET Core 2.0

Technologies discussed:

ASP.NET Core 2.0, .NET Core Project Templates, Razor Pages

look at the project's contents to get a better understanding of how ASP.NET Core works.

Dependencies, Sources and Resources

The first file to look at is the project file itself—the .csproj file. This file tells MSBuild how to build the project—which packages it depends on, what version of .NET Core to target, and so forth. If you've looked at .csproj files in the past, you'll notice this project file is much smaller. A lot of effort has gone into making .csproj files shorter and more human-readable. One notable change that helped with shrinking the project file is that source files no longer need to be listed explicitly. Instead, the .NET Core SDK will automatically compile any .cs files next to the project file or in any directories under the .csproj's directory. Similarly, any .resx files found will be embedded as resources. If you'd prefer to not have all of these .cs files compiled, you can either remove them from the Compile ItemGroup or disable default compile items entirely by setting the EnableDefaultCompileItems property to false.

The .NET version that the app should run against is specified by the <TargetFramework> element. This is set to netcoreapp2.0 in order to take advantage of new ASP.NET Core 2.0 features and the much larger .NET Core 2.0 surface area.

The <PackageReference> element in the middle of the .csproj file indicates a NuGet package on which the project depends. You'll notice in ASP.NET Core 2.0 that you now include just one metapackage by default (Microsoft.AspNetCore.All). This package includes all the other Microsoft.AspNetCore packages in one succinct reference and makes ASP.NET Core 2.0 project files much smaller than project files from previous versions. Additional NuGet dependencies can be added by adding more <PackageReference> elements, by using the Visual Studio NuGet Package management UI, or with the .NET CLI dotnet add command.

If you were using an SPA template (angular, react or reactredux), there would also be custom targets defined in the .csproj file to make sure that Webpack was run when building the project.

Creating and Running the Web Host

Program.cs represents the entry point of the application. ASP.NET Core apps are console applications so, like all console apps, they have a Main method that starts when the app executes.

The contents of the ASP.NET Core 2.0 templates' Main methods are pretty simple—they create an IWebHost object and call Run on it.

If you previously used ASP.NET Core 1.0, you'll notice that this file is a bit simpler than the program.cs from those older templates. The reason for this is the new WebHost.CreateDefaultBuilder method. Previously, the ASP.NET Core app's Main method would configure a WebHostBuilder in order to create the IWebHost instance. This configuration included steps such as specifying a Web server, setting the content root path and enabling IIS integration.

The new CreateDefaultBuilder method simplifies things by creating a ready-to-go IWebHost with most common configuration already done. In addition to specifying the items listed previously, CreateDefaultBuilder also takes care of some setup that was previously handled in Startup.cs (setting up configuration information and registering default logging providers). Because ASP.NET Core is open source, you can see all the details of what CreateDefaultBuilder is doing by viewing its source on GitHub (bit.ly/2uR1Dar) if you're interested.

```
Templates                                 Short Name      Language          Tags
----------------------------------------------------------------------------------------------------
Console Application                       console         [C#], F#, VB      Common/Console
Class library                             classlib        [C#], F#, VB      Common/Library
Unit Test Project                         mstest          [C#], F#, VB      Test/MSTest
xUnit Test Project                        xunit           [C#], F#, VB      Test/xUnit
ASP.NET Core Empty                        web             [C#]              Web/Empty
ASP.NET Core Web App (Model-View-Controller) mvc          [C#], F#          Web/MVC
ASP.NET Core Web App (Razor Pages)        razor           [C#]              Web/MVC/Razor Pages
ASP.NET Core with Angular                 angular         [C#]              Web/MVC/SPA
ASP.NET Core with React.js                react           [C#]              Web/MVC/SPA
ASP.NET Core with React.js and Redux      reactredux      [C#]              Web/MVC/SPA
ASP.NET Core Web API                      webapi          [C#]              Web/WebAPI
Nuget Config                              nugetconfig                       Config
Web Config                                webconfig                         Config
Solution File                             sln                               Solution
Razor Page                                page                              Web/ASP.NET
MVC ViewImports                           viewimports                       Web/ASP.NET
MVC ViewStart                             viewstart                         Web/ASP.NET
```

Figure 1 **New .NET Core Project Templates**

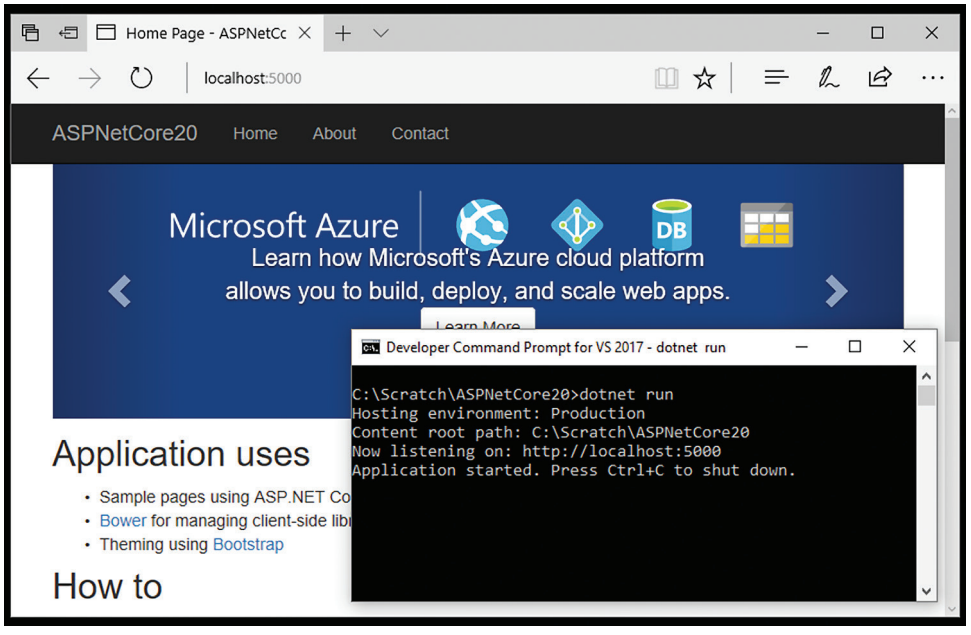


Figure 2 **A Simple ASP.NET Core App Running**

Let's briefly take a look at the most important calls made in CreateDefaultBuilder and their purpose. Although these are all made for you (by CreateDefaultBuilder), it's still good to understand what's happening behind the scenes.

UseKestrel specifies that your application should use Kestrel, a libuv-based cross-platform Web server. The other option here would be to use HttpSys as the Web server (UseHttpSys). HttpSys is supported only on Windows (Windows 7/2008 R2 and later), but has the advantages of allowing Windows authentication and being safe to run directly exposed to the Internet (Kestrel, on the other hand, should sit behind a reverse proxy like IIS, Nginx or Apache if it will receive requests from the Internet).

UseContentRoot specifies the root directory for the application where ASP.NET Core will find site-wide content like config files. Note that this is not the same as the Web root (where static files will be served from), though by default the Web root is based on the content root ([ContentRoot]/wwwroot).

ConfigureAppConfiguration creates the configuration object the app will use to read settings at run time. When called from CreateDefaultBuilder, this will read application configuration settings from an appsettings.json file, an environment-specific .json file (if one exists), environment variables and command-line arguments. If in a development environment, it will also use user secrets. This method is new in ASP.NET Core 2.0 and I'll discuss it in more detail later.

ConfigureLogging sets up logging for the application. When called from CreateDefaultBuilder, console and debug logging providers are added. Like ConfigureAppConfiguration, this method is new and is discussed more later.

UseIISIntegration configures the application to run in IIS. Note that UseKestrel is still needed. IIS is acting as the reverse proxy and Kestrel is still used as the host. Also, UseIISIntegration won't have any effect if the app isn't running behind IIS, so it's safe to call even if the app will be run in non-IIS scenarios.

In many cases, the default configuration provided by CreateDefaultBuilder will be sufficient. All that's needed beyond that call is to specify the Startup class for your application with a call to UseStartup<T>, where T is the Startup type.

If CreateDefaultBuilder doesn't meet your scenario's needs, you should feel free to customize how the IWebHost is being created. If you need to make only small tweaks, you can call CreateDefaultBuilder and then modify the WebHostBuilder that's returned (perhaps by calling ConfigureAppConfiguration again, for example, to add more configuration sources). If you need to make larger changes to the IWebHost, you can skip calling CreateDefaultBuilder completely and just construct a WebHostBuilder yourself, as you would have with ASP.NET Core 1.0 or 1.1. Even if you go this route, you can still take advantage of the new ConfigureAppConfiguration and ConfigureLogging methods. More details on Web host configuration are available at bit.ly/2uuSwwM.

ASP.NET Core Environments

A couple of actions CreateDefaultBuilder takes depend on in which environment your ASP.NET Core application is running. The concept of environments isn't new in 2.0, but is worth briefly reviewing because it comes up frequently.

In ASP.NET Core, the environment an application is running in is indicated by the ASPNETCORE_ENVIRONMENT environment variable. You can set this to any value you like, but the values Development, Staging and Production are typically used. So, if you set the ASPNETCORE_ENVIRONMENT variable to Development prior to calling dotnet run (or if you set that environment variable in a launchSettings.json file), your app will run in Development mode (instead of Production, which is the default without any variables set). This value is used by several ASP.NET Core features (I'll reference it when discussing Configuration and Logging, later) to modify runtime behavior and can be accessed in your own code using the IHostingEnvironment service. More information on ASP.NET Core environments is available in the ASP.NET Core docs (bit.ly/2elCDMF).

ASP.NET Core Configuration

ASP.NET Core uses the Microsoft.Extensions.Configuration package's IConfiguration interface to provide runtime configuration settings. As mentioned previously, CreateDefaultBuilder will read settings from .json files and environment variables. The configuration system is extensible, though, and can read configuration information from a wide variety of providers (.json files, .xml files, .ini files, environment variables, Azure Key Vault and so forth.).

When working with IConfiguration and IConfigurationBuilder objects, remember that the order the providers are added is important. Later providers can override settings from previous providers, so you'll want to add common base providers first and then, later, add environment-specific providers that might override some of the settings.

Configuration settings in ASP.NET Core are hierarchical. In the new project you created, for example, appsettings.json (see **Figure 3**) contains a top-level Logging element with sub-settings underneath it. These settings indicate the minimum priority of messages to log (via the "LogLevel" settings) and whether the app's logical scope at the time the message is logged should be recorded (via IncludeScopes). To retrieve nested settings like these, you can either use the IConfiguration.GetSection method to retrieve a single section of the configuration or specify the full path of a particular setting, delimited with colons. So, the value of IncludeScopes in the project could be retrieved as:

```
Configuration["Logging:IncludeScopes"]
```

When defining configuration settings with environment variables, the environment variable name should include all levels of the hierarchy and can be delimited with either a colon (:) or double underscore (__). For example, an environment variable called Logging__IncludeScopes would override the IncludeScopes setting of the example file in **Figure 3**, assuming that the environment variable provider is added after the settings file, like it is in the CreateDefaultBuilder case.

Because WebHost.CreateDefaultBuilder is reading configuration from both appsettings.json and environment-specific .json files, you'll notice that logging behavior changes when you change environments (appsettings.Development.json overrides the default appsettings.json's LogLevel settings with more verbose "debug" and "information" levels). If you set the environment to Development prior to calling dotnet run, you should notice a fair bit of logging

DevExpress DXperience 17.1 | from $1,439.99

DevExpress

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

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

LEADTOOLS Medical Imaging SDKs V19 | from $4,995.00 SRP

LEADTOOLS
THE WORLD LEADER IN IMAGING SDKs

Powerful DICOM, PACS, and HL7 functionality.

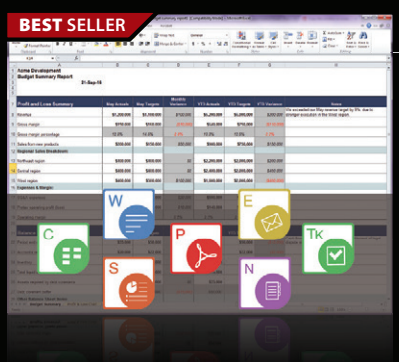
- Load, save, edit, annotate & display DICOM Data Sets with support for the latest specifications
- High-level PACS Client and Server components and frameworks
- OEM-ready HTML5 Zero-footprint Viewer and DICOM Storage Server apps with source code
- Medical-specific image processing functions for enhancing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

Help & Manual Professional | from $586.04

ec software

Help and documentation for .NET and mobile applications.

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



Aspose.Total for .NET | from $2,939.02

ASPOSE
File Format APIs

Create, Edit & Manipulate Word, Excel, PDF and PowerPoint Files in your .NET apps.

- Aspose.Total for .NET contains every Aspose .NET API in one complete package
- Also works with MS Visio, MS OneNote, MS Project, Email, Images, Barcodes, OCR & OMR
- Mail Merge, Document Assembly, Text Extraction, PST & OST Creation and File Conversion
- Create and Recognize Barcodes, Control Text Formatting, Paragraphs, Images and Tables
- Now includes the new Aspose.CAD and Aspose.3D APIs

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

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

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

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

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

GSA Schedule Contract GS-35F-0188R

happening to the console (which is great because it's useful for debugging). On the other hand, if you set the environment to Production, you won't get any console logging except for warnings and errors (which is also great because console logging is slow and should be kept to a minimum in production).

If you have experience with ASP.NET Core 1.0 and 1.1, you might notice that the ConfigureAppConfiguration method is new to 2.0. Previously, it was common to create an IConfiguration as part of the Startup type's creation. Using the new ConfigureAppConfiguration method is a useful alternative because it updates the IConfiguration object stored in the application's dependency injection (DI) container for easy future retrieval and makes configuration settings available even earlier in your application's lifetime.

ASP.NET Core Logging

As with configuration setup, if you're familiar with earlier versions of ASP.NET Core you might remember logging setup being done in Startup.cs instead of in Program.cs. In ASP.NET Core 2.0, logging setup can now be done when building an IWebHost via the ConfigureLogging method.

It's still possible to set up logging in Startup (using services.AddLogging in Startup.ConfigureServices), but by configuring logging at Web host-creation time, the Startup type is streamlined and logging is available even earlier in the app startup process.

Also like configuration, ASP.NET Core logging is extensible. Different providers are registered to log to different endpoints. Many providers are available immediately with a reference to Microsoft.AspNetCore.All, and even more are available from the .NET developer community.

As can be seen in WebHost.CreateDefaultBuilder's source code, logging providers can be added by calling provider-specific extension methods like AddDebug or AddConsole on ILoggingBuilder. If you use WebHost.CreateDefaultBuilder but still want to register logging providers other than the default Debug and Console ones, it's possible to do so with an additional call to ConfigureLogging on the IWebHostBuilder returned by CreateDefaultBuilder.

Once logging has been configured and providers registered, ASP.NET Core will automatically log messages regarding its work to process incoming requests. You can also log your own diagnostic messages by requesting an ILogger object via dependency injection (more on this in the next section). Calls to ILogger.Log and level-specific variants (like LogCritical, LogInformation and so on) are used to log messages.

The Startup Type

Now that you've looked at Program.cs to see how the Web host is created, let's jump over to Startup.cs. The Startup type that your app should use is indicated by the call to UseStartup when creating the IWebHost. Not a lot has changed in Startup.cs in ASP.NET Core 2.0 (except for it becoming simpler because logging and configuration are set up in Program.cs), but I'll briefly review the Startup type's two important methods because they're so central to an ASP.NET Core app.

Dependency Injection The Startup type's ConfigureServices method adds services to the application's dependency injection container. All ASP.NET Core apps have a default dependency injection container to store services for later use. This allows services to be made available without tight coupling to the components that depend on them. You've already seen a couple examples of this—both ConfigureAppConfiguration and ConfigureLogging will add services to the container for use later in your application. At run time, if an instance of a type is called for, ASP.NET Core will automatically retrieve the object from the dependency injection container if possible.

For example, your ASP.NET Core 2.0 project's Startup class has a constructor that takes an IConfiguration parameter. This constructor will be called automatically when your IWebHost begins to run. When that happens, ASP.NET Core will supply the required IConfiguration argument from the dependency injection container.

As another example, if you want to log messages from a Razor Page, you can request a logger object as a parameter to the page model's constructor (like how Startup requests an IConfiguration object) or in cshtml with the @inject syntax, as the following shows:

```
@using Microsoft.Extensions.Logging
@inject ILogger<Index_Page> logger

@functions {
  public void OnGet()
    {
      logger.LogInformation("Beginning GET");
    }
}
```

Something similar could be done to retrieve an IConfiguration object or any other type that has been registered as a service. In this way, the Startup type, Razor Pages, controllers and so forth, can loosely depend on services provided in the dependency injection container.

As mentioned at the beginning of this section, services are added to the dependency injection container in the Startup.ConfigureServices method. The project template you used to create your application already has one call in ConfigureServices: services.AddMvc. As you might guess, this registers services needed by the MVC framework.

Another type of service that's common to see registered in the ConfigureServices method is Entity Framework Core. Although it's not used in this sample, apps that make use of Entity Framework Core typically register the DbContexts needed for working with Entity Framework models using calls to services.AddDbContext.

You can also register your own types and services here by calling services.AddTransient, services.AddScoped or services.AddSingleton (depending on the lifetime required for dependency injection-provided objects). Registering as a singleton will result in a single instance of the service that's returned every time its

Figure 3 **ASP.NET Core Settings File**

```
{
  "Logging": {
    "IncludeScopes": false,
    "Debug": {
      "LogLevel": {
        "Default": "Warning"
      }
    },
    "Console": {
      "LogLevel": {
        "Default": "Warning"
      }
    }
  }
}
```

LIVE! 360
TECH EVENTS WITH PERSPECTIVE

LIVE! 360 AGENDA-AT-A-GLANCE

Visual Studio LIVE!

VISUAL STUDIO LIVE! TRACKS

ALM / DevOps | Cloud Computing | Native Client | Software Practices | Visual Studio / .NET Framework | Web Client | Web Server

SQL Server LIVE!

SQL SERVER LIVE! TRACKS

BI, Big Data, Data Analytics, and Data Visualization | SQL Server Administration & Maintenance | SQL Server in the Cloud | SQL Server for Developers

START TIME	END TIME	Visual Studio Live!	SQL Server Live!
9:00 AM	6:00 PM	**NEW! Visual Studio Live! Full Day Hands-On Labs: Sunday, November 12, 2017** (additional fee applies) **VSS01** Full Day Hands-On Lab: Busy Developer's HOL on Angular - *Ted Neward* **VSS02** Full Day Hands-On Lab: From 0-60 in a Day with Xamarin and Xamarin.Forms - *Roy Cornelissen*	**NEW! SQL Server Live! Full Day Hands-On Labs: Sunday, November 12, 2017** (add **SQS01** Full Day Hands-On Lab: Developer Dive into SQL Server 2016 - *Leonard Lobel* **SQS02** Full Day Hands-On Lab: E SQL Server Availability Option
2:00 PM	7:00 PM	Pre-Conference Registration	Pre-Co
START TIME	END TIME	**Visual Studio Live! Pre-Conference Workshops: Monday, November 13, 2017**	**SQL Server Live! Pre-Conference Workshops: Monday, November 13, 2**
8:30 AM	5:30 PM	**VSM01** Workshop: Distributed Cross-Platform Application Architecture - *Jason Bock & Rockford Lhotka* **VSM02** Workshop: Artificial Intelligence or DevOps?? - *Brian Randell* **VSM03** Workshop: Service Oriented Technologies: Designing, Developing, & Implementing WCF and the Web API - *Miguel Castro*	**SQM01** Workshop: The Modern Data Warehouse - *Bradley Ball & Joshua Luedeman* **SQM02** Workshop: The Secret Toolb Tuning and Optimizing Queries
6:30 PM	8:00 PM	Dine-A-Round Dinner @ Universal CityWalk	Dine-A-Round
START TIME	END TIME	**Visual Studio Live! Day 1: Tuesday, November 14, 2017**	**SQL Server Live! Day 1: Tuesday, November 14, 2017**
8:00 AM	9:00 AM	**VISUAL STUDIO LIVE! KEYNOTE: To Be Announced**	**SQL SERVER LIVE! KEYNOTE:** - *Charles Sterling, Senior Program Manager, Microsoft*
9:15 AM	10:30 AM	**VST01** Front-end Web Development in 2017 for the Front-endally Challenged Microsoft Developer - *Chris Klug* **VST02** Go Mobile with C#, Visual Studio, and Xamarin - *James Montemagno* **VST03** Microservices with Azure Container Service & Service Fabric - *Vishwas Lele* **VST04** What's New in Visual Studio 2017 - *Robert Green*	**SQT01** Business Intelligence/Analytics in SQL Server 2016 - *Thomas LeBlanc* **SQT02** Is SQL Server Making You Wait? Discover the Top 10 Wait Types Every DBA Should Know - *Janis Griffin* **SQT03** Intro Cosmos DB
10:30 AM	11:00 AM	Networking Break • Visit the EXPO	Networking Br
11:00 AM	12:00 PM	**LIVE! 360 KEYNOTE: To Be Announced**	**LIVE! 360 KEYNO**
12:00 PM	12:45 PM	Lunch • Visit the EXPO	Lunch • V
12:45 PM	1:30 PM	Dessert Break • Visit the EXPO	Dessert Brea
1:30 PM	2:45 PM	**VST05** ASP.NET Core MVC—What You Need to Know - *Philip Japikse* **VST06** Optimizing and Extending Xamarin.Forms Mobile Apps - *James Montemagno* **VST07** Tactical DevOps with Visual Studio Team Services - *Brian Randell* **VST08** To Be Announced	**SQT04** What's New for Developers in SQL Server 2016 - *Leonard Lobel* **SQT05** Loading a Data Warehouse Using SSIS - *Tim Mitchell* **SQT06** Turbo Everybod - *Pra*
2:45 PM	3:15 PM	Networking Break • Visit the EXPO	Networking Br
3:15 PM	4:30 PM	**VST09** Angular(2)—The 75-Minute Crash Course - *Chris Klug* **VST10** Building a Cross-Platform Mobile App Backend in the Cloud - *Nick Landry* **VST11** Database Lifecycle Management and the SQL Server Database - *Brian Randell* **VST12** Top 10 Entity Framework Core Features Every Developer Should Know - *Philip Japikse*	**SQT07** SQL Server Analysis Service: Cube Versus Tabular - *Thomas LeBlanc* **SQT08** Advanced SSIS Package Authoring with Biml - *Tim Mitchell* **SQT09** To
4:40 PM	5:00 PM	**VST13** Fast Focus: Aurelia vs. Just Angular - *Chris Klug* **VST14** Fast Focus: Tips & Tricks for Xamarin Development - *James Montemagno* **VST15** Fast Focus on Azure Functions **VST16** Fast Focus: Busting .NET Myths - *Jason Bock*	**SQT10** Fast Focus: Using Polybase to Analyze Data from Other Data Sources, Including Hadoop Data - *Ginger Grant* **SQT11** Fast Focus: A Brave New World—SQL Server on Linux - *Joseph D'Antoni* **SQT12** Fast Data Warehous - *Kar*
5:10 PM	5:30 PM	**VST17** Fast Focus: Web Security 101 - *Brock Allen* **VST18** Fast Focus: Cross-Platform Code Reuse - *Rockford Lhotka* **VST19** Fast Focus: Exploring Microservices in a Microsoft Landscape - *Marcel de Vries* **VST20** Fast Focus: Dependency Injection in 20 Minutes - *Miguel Castro*	**SQT13** Fast Focus: Introduction to Data Science with Azure ML - *Bradley Ball* **SQT14** Fast Focus: Python Essentials for C# Devs - *Jen Stirrup* **SQT15** Fast Database—A - *Tim*
5:30 PM	7:30 PM	Exhibitor Reception	Exhibito
START TIME	END TIME	**Visual Studio Live! Day 2: Wednesday, November 15, 2017**	**SQL Server Live! Day 2: Wednesday, November 15, 2017**
8:00 AM	9:15 AM	**VSW01** User Authentication for ASP.NET Core MVC Applications - *Brock Allen* **VSW02** Cloud Oriented Programming - *Vishwas Lele* **VSW03** Overcoming the Challenges of Mobile Development in the Enterprise - *Roy Cornelissen* **VSW04** Building Apps with Microsoft Graph and Visual Studio - *Robert Green*	**SQW01** Power BI: What Have You Done for Me Lately? - *Andrew Brust* **SQW02** Getting Started with Query Tuning - *Joseph D'Antoni* **SQW03** Azu and - *Gra*
9:30 AM	10:45 AM	**VSW05** Building AngularJS Component-Based Applications - *Miguel Castro* **VSW06** Building Modern Web Apps with Azure - *Eric D. Boyd* **VSW07** Creating a Release Pipeline with Team Services - *Esteban Garcia* **VSW08** Bots are the New Apps: Building Bots with ASP.NET Web API & Language Understanding - *Nick Landry*	**SQW04** Graph DB Support in SQL Server 2017 - *Karen Lopez* **SQW05** HA and DR in the Cloud--Building Robust SQL Server Solutions on Azure - *Joseph D'Antoni* **SQW06** Car in Micros - *Thom*
10:45 AM	11:30 AM	Networking Break • Visit the EXPO	Networking Br
11:30 AM	12:30 PM	**LIVE! 360 KEYNOTE: To Be Announced**	**LIVE! 360 KEYNO**
12:30 PM	1:30 PM	Birds-of-a-Feather Lunch	Birds-of-a
1:30 PM	2:00 PM	Dessert Break • Visit the EXPO	Dessert Brea
2:00 PM	3:15 PM	**VSW09** Securing Web APIs in ASP.NET Core - *Brock Allen* **VSW10** A/B Testing, Canary Releases and Dark Launching, Implementing Continuous Delivery on Azure - *Marcel de Vries* **VSW11** The Zen of UI Automation Testing **VSW12** To Be Announced	**SQW07** Big Data Technologies: What, Where and How to Run Them on Azure - *Andrew Brust* **SQW08** Introduction to R - *Bradley Ball* **SQW09** To
3:15 PM	4:00 PM	Networking Break • Visit the EXPO • Expo Raffle	Networking Break • V
4:00 PM	5:15 PM	**VSW13** Build Object-Oriented Enterprise Apps in JavaScript with TypeScript **VSW14** Lock the Doors, Secure the Valuables, and Set the Alarm - *Eric D. Boyd* **VSW15** Unit Testing & Test-Driven Development (TDD) for Mere Mortals - *Benjamin Day* **VSW16** PowerApps, Flow, and Common Data Service: Empowering Businesses with the Microsoft Business Application Platform - *Charles Sterling*	**SQW10** Data Lake Analytics with U-SQL - *Ginger Grant* **SQW11** Big Data Smorgasbord with R in Hadoop and SQL Server - *Jen Stirrup* **SQW12** Explor - *Gra*
8:00 PM	10:00 PM	Live! 360 Dessert Luau	Live! 360
START TIME	END TIME	**Visual Studio Live! Day 3: Thursday, November 16, 2017**	**SQL Server Live! Day 3: Thursday, November 16, 2017**
8:00 AM	9:15 AM	**VSH01** HTTP/2: What You Need to Know - *Robert Boedigheimer* **VSH02** PowerApps and Flow Part II: Package, Embed, and Extend Your Applications - *Manas Maheshwari & Pratap Ladhani* **VSH03** Exploring C# 7 New Features - *Adam Tuliper* **VSH04** Top 10 Ways to Go from Good to Great Scrum Master - *Benjamin Day*	**SQH01** Top Five SQL Server Query Tuning Tips - *Janis Griffin* **SQH02** Windows PowerShell Jumpstart for the SQL Server DBA - *Edwin M. Sarmiento* **SQH03** In Azure D - *Tim*
9:30 AM	10:45 AM	**VSH05** ASP.NET Tag Helpers - *Robert Boedigheimer* **VSH06** Storyboarding 101 - *Billy Hollis* **VSH07** .NET Standard—From Noob to Ninja - *Adam Tuliper* **VSH08** Devs vs. Ops: Making Friends with the Enemy - *Damian Brady*	**SQH04** Upgrading to SQL Server 2016 - *Thomas LaRock* **SQH05** A Database Designer's Favorite SQL Server Security & Privacy Features - *Karen Lopez* **SQH06** Mat Databases L - *Gra*
11:00 AM	12:00 PM	**Visual Studio Live! Panel**: To Be Announced - *Brian Randell (Moderator), Damian Brady, Jeremy Clark, Esteban Garcia, Billy Hollis, & Adam Tuliper*	**SQL Server Live! Panel**: In with The New, And The Old Too: The Database New World O - *Andrew Brust (Moderator), Ginger Grant, Thomas LaRock, Leonard Lobel, Karen Lopez, and J*
12:00 PM	1:00 PM	Lunch on the Lanai	Lunch o
1:00 PM	2:15 PM	**VSH09** I See You: Watching the User with Reactive Forms - *Deborah Kurata* **VSH10** Continuous Integration and Delivery for Mobile Applications using VSTS - *Kevin Ford* **VSH11** Deploying Straight to Production: A Guide to the Holy Grail - *Damian Brady* **VSH12** Design Patterns: Not Just for Architects - *Jeremy Clark*	**SQH07** T-SQL User-Defined Functions, or: How to Kill Performance in One Easy Step - *Hugo Kornelis* **SQH08** SQL Server 2016 Database Administration for the Non-DBA - *Denny Cherry* **SQH09** Ligh Analytics with A - *Jen*
2:30 PM	3:45 PM	**VSH13** Angular Routing - *Deborah Kurata* **VSH14** XAML Inception—Deep Composition for Better UI - *Billy Hollis* **VSH15** Application Insights: Measure Your Way to Success - *Esteban Garcia* **VSH16** DI Why? Getting a Grip on Dependency Injection - *Jeremy Clark*	**SQH10** Adaptive Query Processing - *Hugo Kornelis* **SQH11** Using SQL Server AlwaysOn Availability Groups - *Denny Cherry* SQH12 Oper in SQL S - *Edwin*
4:00 PM	5:00 PM	**Next? Visual Studio Live! Networking Event** - *Brian Randell (Moderator), Damian Brady, Jeremy Clark, Esteban Garcia, Billy Hollis, & Deborah Kurata*	**Next? SQL Server Live! Networking Event** - *Thomas LaRock (Moderator), Bradley Ball, Andrew Brust, Denny Cherry, and Karen Lo*
START TIME	END TIME	**Visual Studio Live! Post-Conference Workshops: Friday, November 17, 2017**	**SQL Server Live! Post-Conference Workshops: Friday, November 17, 2**
8:00 AM	5:00 PM	**VSF01** Workshop: Angular Fundamentals - *John Papa* **VSF02** Workshop: Building, Running & Continuously Deploying Microservices with Docker Containers on Azure - *Marcel de Vries & Rene van Osnabrugge*	**SQF01** Workshop: Big Data, BI, and Analytics on The Microsoft Stack- *Andrew Brust* **SQF02** Workshop: Database Admini Database Administrator - De

Speakers and sessions subject to change

ROYAL PACIFIC RESORT
AT UNIVERSAL ORLANDO
NOVEMBER 12-17

2017 Orlando

TECHMENTOR

TECHMENTOR TRACKS

SQL Server Performance Tuning and Optimization	Client	DevOps	Infrastructure	IT Soft Skills	Security	Server / Datacenter	The Real Cloud

NEW! TechMentor Full Day Hands-On Labs: Sunday, November 12, 2017 (additional fee applies)

- **TMS01** Hands-On Lab: Ethical Hacking with Kali Linux - *Mike Danseglio & Avril Salter*
- **TMS02** Hands-On Lab: Advanced Windows Troubleshooting for IT/Ops Pros - *Bruce Mackenzie-Low*

Pre-Conference Registration

TechMentor Pre-Conference Workshops: Monday, November 13, 2017

- **TMM01** Workshop: Learn the latest and Greatest Updates to the Azure Platform IaaS and PaaS Services - *Peter De Tender*
- **TMM02** Workshop: Windows Security—How I Do It! - *Sami Laiho*

Dinner @ Universal CityWalk

TechMentor Day 1: Tuesday, November 14, 2017

TECHMENTOR KEYNOTE: To Be Announced

- **TMT01** Right Tool for the Right Job! Group Policy, SCCM, MDM, Azure AD, and More. Do Management Right! - *Jeremy Moskowitz*
- **TMT02** Secure Access Everywhere! Implementing DirectAccess with Windows Server 2016 - *Richard Hicks*
- **TMT03** Learn What It Takes to Build a Cockpit Alike View for Monitoring all of Your Azure Resources - *Peter De Tender*

Networking Break • Visit the EXPO

- **TMT04** Keepin' The Bad Guys Out of Windows 10 with Microsoft's In-box (and 3rd party) Tools - *Jeremy Moskowitz*
- **TMT05** Become a Windows Server 2016 Cluster Master in Five Easy Architectures - *Greg Shields*
- **TMT06** An Absolute Beginners Guide to Storage Spaces Direct (S2D) - *Dave Kawula*

Networking Break • Visit the EXPO

- **TMT07** Pitfalls and Problems: Windows 10 and Server 2016 - *Jeremy Moskowitz*
- **TMT08** Hardware, Camtasia, and a Storyline: Creating Your Own User Training - *Greg Shields*
- **TMT09** To Be Announced
- **TMT10** Fast Focus: New Ways to Install Windows 10 Without Imaging, Is Task Sequence Dead? - *Petri Paavola*
- **TMT11** Fast Focus: Preparing for the Windows Server 2016 MCSA—The Expert Tips - *Greg Shields*
- **TMT12** Fast Focus: Mythbusters—Windows Pagefile-Settings - *Sami Laiho*
- **TMT13** Fast Focus: System Center Configuration Manager Current Branch: How to Survive with Fast Release Cadence - *Panu Saukko*
- **TMT14** Fast Focus: Recovering Your On-premises Workloads to Azure in 20 minutes—with Nothing But Demos - *Peter De Tender*
- **TMT15** Fast Focus: Mythbusters—Tweaking Windows Performance - *Sami Laiho*

Exhibitor Reception

TechMentor Day 2: Wednesday, November 15, 2017

- **TMW01** The Network is Slow! Or is it? Network Troubleshooting for Windows Administrators - *Richard Hicks*
- **TMW02** Configuring Active Directory Certificate Services for DSC Credential Encryption - *Melissa Januszko*
- **TMW03** Bypass Every Security Technology with Social Hacking: A Beginner's Toolkit - *Dale Meredith*
- **TMW04** Intelligent DNS for Modern Networks: Introducing DNS Policies in Windows Server 2016 - *Richard Hicks*
- **TMW05** Make Your PowerShell Scripts Bulletproof with Pester - *Melissa Januszko*
- **TMW06** Pen Testing with PowerShell: Cmdlets That'll Pwn Your Network - *Dale Meredith*

Networking Break • Visit the EXPO

Birds-of-a-Feather Lunch

Dessert Break • Visit the EXPO

- **TMW07** Automated Troubleshooting Techniques in Enterprise Domains - *Petri Paavola*
- **TMW08** Create a Customized Office 365 Software Deployment Package - *John O'Neill, Sr.*
- **TMW09** To Be Announced

Networking Break • Visit the EXPO • Expo Raffle

- **TMW10** How to Manage Windows 10 Devices with System Center Configuration Manager Current Branch - *Panu Saukko*
- **TMW11** Putting the Windows Assessment and Deployment Kit to Work - *John O'Neill, Sr.*
- **TMW12** Controlling Your Azure Spend - *Timothy Warner*

Live! 360 Dessert Luau

TechMentor Day 3: Thursday, November 16, 2017

- **TMH01** Fully Integrated Azure Resource Manager Deployments - *Neil Peterson*
- **TMH02** Keep Up with NOW! Automate Rebuilding Your (Home) Lab—On Steroids - *Sven van Rijen*
- **TMH03** Regular Expressions Jump Start - *Timothy Warner*
- **TMH04** Automating Azure with PowerShell—Beyond the Azure PowerShell Module - *Neil Peterson*
- **TMH05** Weakest Link of Office 365 Security - *Nestori Syynimaa*
- **TMH06** Git and GitHub Jump Start - *Timothy Warner*

TechMentor Panel: The Future of Windows Management - *Moderator: Sami Laiho*

Lunch on the Lanai

- **TMH07** Moving from Active Directory to Azure Active Directory Quick Wins and Common Challenges - *Tarek Dawoud & Mark Morowczynski*
- **TMH08** Nine O365 Security Issues Microsoft Probably Has Not Told You (and You Probably Don't Want to Know) - *Nestori Syynimaa*
- **TMH09** PowerShell Scripting Secrets - *Jeffery Hicks*
- **TMH10** Deep Dive on SaaS Apps with Azure Active Directory - *Tarek Dawoud & Mark Morowczynski*
- **TMH11** Starting Your Own Gig: Three Steps to Launching Your Own Business - *Nick Cavalancia*
- **TMH12** In-Depth Introduction to Docker - *Neil Peterson*

Next? TechMentor Networking Event - *Moderator: Sami Laiho*

TechMentor Post-Conference Workshops: Friday, November 17, 2017

- **TMF01** Workshop: Managing Windows 10 from the Cloud - *Panu Saukko & Petri Paavola*
- **TMF02** Workshop: Advanced Desired State Configuration - *Jeffery Hicks*

Office & SharePoint LIVE!

OFFICE & SHAREPOINT LIVE! TRACKS

SharePoint On-Premises Infrastructure Management and Administration	SharePoint, Office 365 and the Cloud	High-Value SharePoint Workloads: Social, Search, BI, Workflow, and Business Process Automation	Developing for Office, Office 365 and SharePoint	Information and Content Management: Documents, Records, and Web

NEW! Office & SharePoint Live! Full Day Hands-On Labs: Sun, Nov 12 (additional fee applies)

- **OSS01** Full Day Hands-On Lab: Search Managers 101 - *Agnes Molnar*
- **OSS02** Full Day Hands-On Lab: Developing Extensions for Microsoft Teams - *Paul Schaeflein*

Pre-Conference Registration

Office & SharePoint Live! Pre-Conference Workshops: Monday, Nov 13, 2017

- **OSM01** Workshop: Configuring Hybrid Workloads for Office 365 & SharePoint - *Matthew McDermott*
- **OSM02** Workshop: Mastering the SharePoint Framework - *Andrew Connell*

Dine-A-Round Dinner @ Universal CityWalk

Office & SharePoint Live! Day 1: Tuesday, November 14, 2017

OFFICE & SHAREPOINT LIVE! KEYNOTE: - *Yina Arenas, Principal Program Manager Lead, Microsoft*

- **OST01** "Quick Win" Features vs. Long Term Strategy in Search - *Agnes Molnar*
- **OST02** TypeScript for SharePoint Developers - *Rob Windsor*

Networking Break • Visit the EXPO

LIVE! 360 KEYNOTE: To Be Announced

Lunch • Visit the EXPO

Dessert Break • Visit the EXPO

- **OST03** Managing and Configuring Office 365 Identity - *Scott Hoag*
- **OST04** To Be Announced

Networking Break • Visit the EXPO

- **OST05** Intelligent Insights and Collaboration in Office 365 - *Agnes Molnar*
- **OST06** To Be Announced - *Yina Arenas*
- **OST07** Fast Focus: 5 Query Rules Every SharePoint Pro Should Know - *Matthew McDermott*
- **OST08** Fast Focus: Using Features to Provision Assets in SharePoint Framework Solutions - *Rob Windsor*
- **OST09** Fast Focus: The 5 PowerShell Tricks for SharePoint You NEED to Know - *Ben Stegink*
- **OST10** Fast Focus: SharePoint Developers Sound Off - *Andrew Connell*

Exhibitor Reception

Office & SharePoint Live! Day 2: Wednesday, November 15, 2017

- **OSW01** What Every IT Pro Needs to Know About SharePoint On-Premises - *Robert Bogue*
- **OSW02** Building Office Add-ins for Outlook with Angular - *Andrew Connell*
- **OSW03** What Every IT Pro Needs to Know About SharePoint Online - *Robert Bogue*
- OSW04 Introduction to the SharePoint Client Object Model and REST API - *Rob Windsor*

Networking Break • Visit the EXPO

LIVE! 360 KEYNOTE: To Be Announced

Birds-of-a-Feather Lunch

Dessert Break • Visit the EXPO

- **OSW05** Managing Office 365 Groups in Office 365 - *Scott Hoag*
- **OSW06** Developing with the SharePoint Framework: Web Parts & Customizing the UI - *Andrew Connell*

Networking Break • Visit the EXPO • Expo Raffle

- **OSW07** Upgrade to SharePoint 2016 - *Matthew McDermott*
- **OSW08** Developing SharePoint Framework Components Using Visual Studio - *Paul Schaeflein*

Live! 360 Dessert Luau

Office & SharePoint Live! Day 3: Thursday, November 16, 2017

- **OSH01** What Every Developer Needs to Know about SharePoint Development Online or On-Prem - *Robert Bogue*
- **OSH02** Feel the Flow—Adding your Custom API to Microsoft Flow - *Paul Schaeflein*
- **OSH03** Office 365 and Hybrid Solutions—What Works for My Organization? - *Scott Hoag*
- **OSH04** BI for the Cobbler's Children: JavaScript Charting and Graphing - *Julie Turner*

Office & SharePoint Live! Panel: To Be Announced – *Andrew Connell & Matthew McDermott (Moderators), Brian Pendergrass*

Lunch on the Lanai

- **OSH05** All about PowerShell for SharePoint - *Ben Stegink*
- **OSH06** Introduction to the Office Dev PnP Core Libraries - *Rob Windsor*
- **OSH07** SharePoint Cloud Hybrid Search—A Field Guide to Effective Planning and Deployments - *Brian Pendergrass*
- **OSH08** Build a Complete Business Solution Using Microsoft Graph API through Client Side Web Parts - *Julie Turner*

Next? Office & SharePoint Live! Networking Event - *Moderators: Andrew Connell & Matthew McDermott*

Office & SharePoint Live! Post-Conference Workshops: Friday, November 17, 2017

- **OSF01** Workshop: Deploying Office 365: Soup to Nuts - *Ben Stegink & Scott Hoag*

ModernApps LIVE!

MODERN APPS LIVE! TRACK

Presented in Partnership with: Magenic

MAL! Pre-Con Workshop: Monday, Nov 13

- **MAM01** Workshop: Building Modern Mobile Apps - *Brent Edwards & Kevin Ford*

MAL! Day 1: Tuesday, Nov 14

MAL! KEYNOTE PANEL: Industry Trends, Technology, and Your Career - *Matt Lockhart (Moderator)*

- **MAT01** Modern App Development: Transform How You Build Web and Mobile Software - *Rockford Lhotka*
- **MAT02** Architecture: The Key to Modern App Success - *Brent Edwards*
- **MAT03** Focus on the User Experience #FTW - *Jim Barrett*
- **MAT04** Fast Focus: Hybrid Web Frameworks - *Allen Conway*
- **MAT05** Fast Focus: Web Assembly - Jason Bock

MAL! Day 2: Wednesday, Nov 15

- **MAW01** Manage Distributed Teams with Visual Studio Team Services and Git - *Brian Randell*
- **MAW02** DevOps, Continuous Integration, the Cloud, and Docker - *Dan Nordquist*
- **MAW03** Security with Speed for Modern Developers - *Michael Lester*
- **MAW04** Coding for Quality and Maintainability - *Jason Bock*

MAL! Day 3: Thursday, Nov 16

- **MAH01** Modern Web Development: Building Server Side Using ASP.NET Core, MVC, Web API, and Azure - *Allen Conway*
- **MAH02** Modern Web Development: Building Client Side Using TypeScript and Angular - *Allen Conway*

MAL! Panel: Mobile Development Technologies - *Rockford Lhotka (Moderator), James Montemagno, Kevin Ford*

- **MAH03** Using All That Data: Power BI to the Rescue - *Scott Diehl*
- **MAH04** Modern Mobile Development: Build a Single App For iOS, Android, and Windows with Xamarin Forms - Kevin Ford

Next? Modern Apps Live! Networking Event - *Moderator: Rockford Lhotka*

MAL! Post-Con Workshop: Friday, Nov 17

- **MAF01** Workshop: Modern App Deep Dive—Xamarin, Responsive Web, UWP - *Kevin Ford, Brent Edwards, Allen Conway*

REGISTER NOW AT LIVE360EVENTS.COM/MSDN

2017 Orlando

ROYAL PACIFIC RESORT
AT UNIVERSAL ORLANDO
NOVEMBER 12-17

REGISTER NOW

REGISTER BY SEPT 15 AND SAVE $400!*

Use promo code L360TIPIN

*Restrictions apply.
See website for details.

CONNECT WITH LIVE! 360

twitter.com/live360
@live360events

facebook.com
Search "Live 360"

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

LIVE! 360
TECH EVENTS WITH PERSPECTIVE

5 GREAT CONFERENCES, 1 GREAT PRICE

Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS

Visual Studio Live!: Code in Paradise at VSLive!™, featuring unbiased and practical development training on the Microsoft Platform.

SQL Server LIVE!
TRAINING FOR DBAs AND IT PROS

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

TECHMENTOR
IN-DEPTH TRAINING FOR IT PROS

TechMentor: This is where IT training meets sunshine, with zero marketing speak on topics you need training on now, and solid coverage on what's around the corner.

Office & SharePoint LIVE!
ON-PREMISE, CLOUD & CROSS-PLATFORM TRAINING

Office & SharePoint Live!: Today, organizations expect people to work from anywhere at any time. Office & SharePoint Live! provides leading-edge knowledge and training to work through your most pressing projects.

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

Modern Apps Live!: Presented in partnership with Magenic, this unique conference delivers on what you need to know to architect, design and build a complete Modern App.

LIVE360EVENTS.COM/MSDN

LIVE! 360
TECH EVENTS WITH PERSPECTIVE

FEATURED SPEAKERS

Live! 360 brings you over 90 speakers; some of the best and brightest experts in the industry. View the full list of speakers at live360events.com

Visual Studio LIVE!


Deborah Kurata

Jason Bock

Billy Hollis

Miguel Castro

Marcel de Vries

Adam Tuliper

Nick Landry

Brian Noyes

John Papa

Brian Randell

SQL Server LIVE!


Andrew Brust

Pinal Dave

Edwin Sarmiento

Leonard Lobel

Janis Griffin

TECHMENTOR


Jeffery Hicks

Avril Salter

Sami Laiho

Tim Warner

Greg Shields

Office & SharePoint LIVE!


Robert Bogue

Rob Windsor

Andrew Connell

Matthew McDermott

Agnes Molnar

Modern Apps LIVE!


Allen Conway

Brent Edwards

Kevin Ford

Scott Diehl

Rockford Lhotka

REGISTER NOW

REGISTER BY SEPT 15 AND SAVE $400!*

Use promo code L360TIPIN

*Restrictions apply.
See website for details.

LIVE360EVENTS.COM/MSDN

CONNECT WITH LIVE! 360

twitter.com/live360
@live360events

facebook.com
Search "Live 360"

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

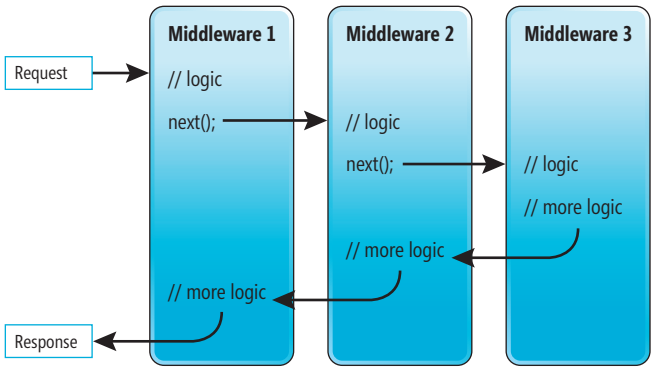


Figure 4 **ASP.NET Core Middleware Processing Pipeline**

type is requested, whereas registering as transient will cause a new instance to be created for each request. Adding as scoped will cause a single instance of a service to be used throughout the processing of a single HTTP request. More details on dependency injection in ASP.NET Core are available at bit.ly/2uqOhDc.

HTTP Request-Processing Pipeline and Middleware The other important method in the Startup type is the Configure method. This is where the heart of the ASP.NET Core application—its HTTP request-processing pipeline—is set up. In this method, different pieces of middleware are registered that will act on incoming HTTP requests to generate responses.

In Startup.Configure, middleware components are added to an IApplicationBuilder to form the processing pipeline. When a request comes in, the first piece of middleware registered will be invoked. That middleware will perform whatever logic it needs to and then either call the next piece of middleware in the pipeline or, if it has completely handled the response, return to the previous piece of middleware (if there was a previous one) so that it can execute any logic necessary after a response has been prepared. This pattern of calling middleware components in order when a request arrives and then in reverse order after it has been handled is illustrated in **Figure 4**.

To take a concrete example, **Figure 5** shows the Configure method from the template project. When a new request comes in, it will go first to either the DeveloperExceptionPage middleware

Figure 5 **ASP.NET Core Startup.Configure Method Sets Up the Middleware Pipeline**

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
  if (env.IsDevelopment())
  {
    app.UseDeveloperExceptionPage();
  }
  else
  {
    app.UseExceptionHandler("/Error");
  }

  app.UseStaticFiles();

  app.UseMvc(routes =>
  {
    routes.MapRoute(
      name: "default",
      template: "{controller=Home}/{action=Index}/{id?}");
  });
}
```

or the ExceptionHandler middleware, depending on your environment (as before, this is configured with the ASPNETCORE_ENVIRONMENT environment variable). These middleware components won't take much action initially, but after subsequent middleware has run and the request is on its way back out of the middleware pipeline, they will watch for and handle exceptions.

Next, StaticFiles middleware will be called, which may serve the request by providing a static file (an image or style sheet, for example). If it does, it will halt the pipeline and return control to the previous middleware (the exception handlers). If the StaticFiles middleware can't provide a response, it will call the next piece of middleware—the MVC middleware. This middleware will attempt to route the request to an MVC controller (or Razor Page) for fulfillment, according to the routing options specified.

The order the middleware components are registered is very important. If UseStaticFiles came after UseMvc, the application would try to route all requests to MVC controllers before checking for static files. That could result in a noticeable perf degrade! If the exception-handling middleware came later in the pipeline, it wouldn't be able to handle exceptions occurring in prior middleware components.

Razor Pages

Besides the .csproj, program.cs, and startup.cs files, your ASP.NET Core project also contains a Pages folder containing the application's Razor Pages. Pages are similar to MVC views, but requests can be routed directly to a Razor Page without the need for a separate controller. This makes it possible to simplify page-based applications and keep views and view models together. The model supporting the page can be included in the cshtml page directly (in a @functions directive), or in a separate code file, which is referenced with the @model directive.

To learn more about Razor Pages, check out Steve Smith's article, "Simpler ASP.NET MVC Apps with Razor Pages," also in this issue.

Wrapping Up

Hopefully this walk-through helps explain how to create a new ASP.NET Core 2.0 Web application and demystifies the contents of the new project templates. I've reviewed the project contents from the streamlined .csproj file, to the application entry point and Web host configuration in Program.cs, to service and middleware registration in Startup.cs.

To keep digging deeper into what's possible with ASP.NET Core, it might be useful to create some new projects using some of the other templates, like the Web API template or perhaps some of the new SPA templates. You may also want to try out deploying your ASP.NET Core app to Azure as an App Service Web App or packaging the application as a Linux or Windows Docker image. And, of course, please check out the full documentation at docs.microsoft.com/aspnet/core for more information on the topics covered in this article and more. ■

Mike Rousos *is a principal software engineer on the .NET Customer Success Team. Rousos has been a member of the .NET team since 2004, working on technologies including tracing, managed security, hosting and, most recently, .NET Core.*

Thanks to the following Microsoft technical experts who reviewed this article: *Glenn Condron and Ryan Nowak*

Simpler ASP.NET MVC Apps with Razor Pages

Steve Smith

Razor Pages are a new feature in ASP.NET Core 2.0. They provide a simpler way to organize code within ASP.NET Core applications, keeping implementation logic and view models closer to the view implementation code. They also offer a simpler way to get started developing ASP.NET Core apps, but that doesn't mean you should dismiss them if you're an experienced .NET developer. You can also use Razor Pages to improve the organization of larger and more complex ASP.NET Core apps.

The Model-View-Controller (MVC) pattern is a mature UI pattern that Microsoft has supported for developing ASP.NET applications since 2009. It offers a number of benefits that can help application developers achieve a separation of concerns, resulting in more maintainable software. Unfortunately, the pattern as implemented in the default project templates often results in a lot of files and folders, which can add friction to development, especially as an application grows. In my September 2016 article, I wrote about using Feature Slices as one approach to address this issue (msdn.com/magazine/mt763233). Razor Pages offer a new and different way to tackle this same problem, especially for scenarios that are conceptually page-based. This approach is especially useful when all you have is a nearly static view, or a simple form that just needs to perform a POST-Redirect-GET. These scenarios are the sweet spot for Razor Pages, which avoid a great deal of the convention required by MVC apps.

This article discusses:

- Getting started using Razor Pages
- Demonstrating with a sample project
- Routing, model binding and handlers
- Using filters
- The Razor Pages architectural pattern

Technologies discussed:

ASP.NET Core 2.0, Razor Pages, Web UI Patterns and Architecture

Code download available at:

bit.ly/2eJ01cS

Getting Started with Razor Pages

To get started using Razor Pages, you can create a new ASP.NET Core Web Application in Visual Studio using ASP.NET Core 2.0, and select the Razor Pages template, as shown in **Figure 1**.

You can achieve the same thing from the dotnet command-line interface (CLI) using:

```
dotnet new razor
```

You'll need to make sure you're running at least version 2.0 of the .NET Core SDK; check with:

```
dotnet --version
```

In either case, if you examine the project produced, you'll see it includes a new folder, Pages, as shown in **Figure 2**.

Notably absent from this template are two folders that are typically associated with MVC projects: Controllers and Views. Razor Pages use the Pages folder to hold all of the pages for the application. You're

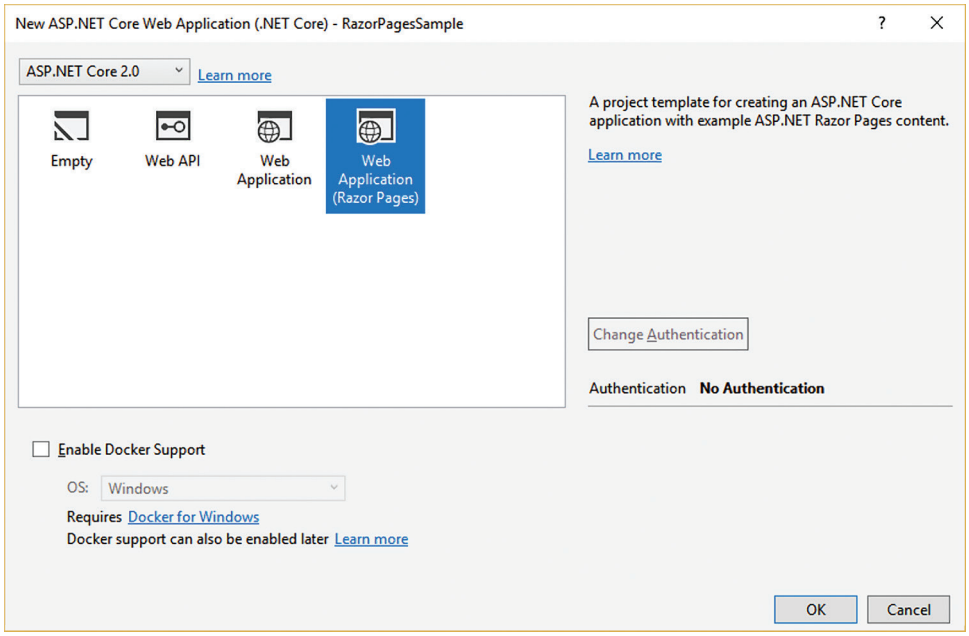


Figure 1 **ASP.NET Core 2.0 Web Application with Razor Pages Template**

free to use folders within the Pages root folder to organize pages in whatever way makes sense for your application. Razor Pages allow developers to combine the code-quality features of the MVC pattern with the productivity benefits of grouping together things that tend to change together.

Note that Pages is part of ASP.NET Core MVC in version 2. You can add support for Pages to any ASP.NET Core MVC app by simply adding a Pages folder and adding Razor Pages files to this folder.

Razor Pages use the folder structure as a convention for routing requests. While the default page in a typical MVC app can be found at "/," as well as "/Home/" and "/Home/Index," the default Index page in an app using Razor Pages will match "/" and "/Index." Using subfolders, it's very intuitive to create different sections of your app, with routes that match accordingly. Each folder can have an Index.cshtml file to act as its root page.

Looking at an individual Page, you'll find there's a new page directive, @page, that's required on Razor Pages. This directive must appear on the first line of the page file, which should use the .cshtml extension. Razor Pages look and behave very similarly to Razor-based View files, and a very simple page can include just HTML:

```
@page
<h1>Hello World</h1>
```

Where Razor Pages shine is in encapsulating and grouping UI details. Razor Pages support inline or separate class-based page models, which can represent data elements the page will display or manipulate. They also support handlers that eliminate the need for separate controllers and action methods.

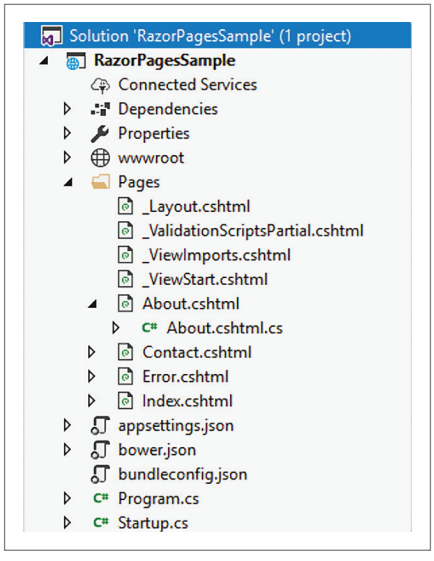


Figure 2 **Razor Pages Project Template Organization**

These features greatly reduce the number of separate folders and files required to work with a given page on a Web app. **Figure 3** compares the folders and files required for a typical MVC-based approach with the Razor Pages approach.

To demonstrate Razor Pages in the context of an ASP.NET Core MVC app, I'm going to use a simple sample project.

A Sample Project

To simulate a project with a little bit of complexity and some different feature areas, I'm going to return to the sample I used for my Feature Slices article. This sample involves viewing and managing a number of different kinds of entities, including ninjas and ninja swords, as well as pirates, plants, and zombies. Imagine the app is a companion for a casual game and helps you manage in-game constructs. Using the typical MVC organizational approach, you'd most likely have many different folders holding controllers, views, viewmodels, and more for each of these kinds of constructs. With Razor Pages, you can create a simple folder hierarchy that maps to your application's URL structure.

In this case, the application has a simple homepage and four different sections, each with its own subfolder under Pages. The folder structure is very clean, with just the homepage (Index.cshtml) and some supporting files in the root of the Pages folder, and the other sections in their own folders, as **Figure 4** shows.

Simple pages often don't need separate page models. For example, the list of ninja swords shown in /Ninjas/Swords/Index.cshtml simply uses inline variables, as **Figure 5** shows.

Variables declared in Razor blocks are in scope on the page; you'll see how you can declare functions and even classes via @functions blocks in the next section. Note the use of the new asp-page tag helper in the link at the bottom of the page. These tag helpers reference pages by their routes, and support absolute and relative paths. In this example, "/Ninjas/Index" could also have been written as "../Index" or even just ".." and it would route to the same Index.cshtml Razor Page in the Ninjas folder. You can also use the asp-page tag helper on <form> elements to specify a form's destination. Because the asp-page tag helpers build on top of the powerful ASP.NET Core routing support, they support many URL generation scenarios beyond simple relative URLs.

Page Models

Razor Pages can support strongly typed page models. You specify the model for a Razor Page with the @model directive (just like a strongly typed MVC View). You can define the model within the Razor Page file, as shown in **Figure 6**.

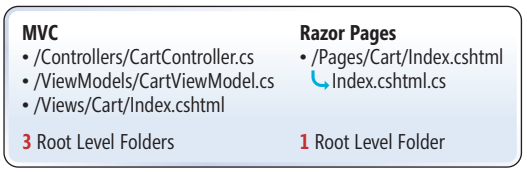


Figure 3 **MVC Folders and Files vs. Razor Pages**

You can also define the page model in a separate codebehind file named Pagename.cshtml.cs. In Visual Studio, files that follow this convention are linked to their corresponding page file, making it easy to navigate between them. The same code shown in the @functions block in **Figure 6** could be placed into a separate file.

There are pros and cons to both approaches for storing page models. Placing page-model logic within the Razor Page itself results in fewer files and allows for the flexibility of runtime compilation, so you can make updates to the page's logic without the need for a full deployment of the app. On the other hand, compilation errors in page models defined within Razor Pages may not be discovered until runtime. Visual Studio will show errors in open Razor files (without actually compiling them). Running the dotnet build command doesn't compile Razor Pages or provide any information about potential errors in these files.

Separate page-model classes offer slightly better separation of concerns, because the Razor Page can focus purely on the template for displaying data, leaving the separate page model to handle the structure of the page's data and the corresponding handlers. Separate codebehind page models also benefit from compile-time error checking and are easier to unit test than inline page models. Ultimately, you can choose whether to use no model, an inline model or separate page models in your Razor Pages.

Routing, Model Binding and Handlers

Two key features of MVC that are typically found within Controller classes are routing and model binding. Most ASP.NET Core MVC apps use attributes to define routes, HTTP verbs and route parameters, using syntax like this:

```
[HttpGet("{id}")]
public Task<IActionResult> GetById(int id)
```

As previously noted, the route path for Razor Pages is convention-based, and matches the page's location within the /Pages folder hierarchy. However, you can support route parameters by adding them to the @page directive. Instead of specifying supported HTTP verbs using attributes, Razor Pages use handlers that follow a naming convention of On*Verb*, where Verb is an HTTP verb like Get, Post and so on. Razor Page handlers behave very similarly to MVC Controller actions, and they use model binding to populate any parameters they define. **Figure 7** shows a sample Razor Page that uses route parameters, dependency injection and a handler to display the details of a record.

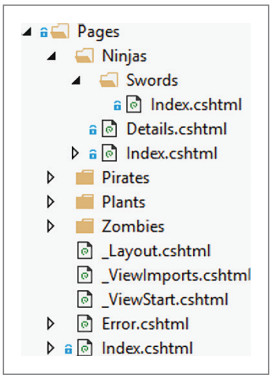


Figure 4 **Folder Organization with Razor Pages**

Pages can support multiple handlers, so you can define OnGet, OnPost and so forth. Razor Pages also introduce a new model-binding attribute, [BindProperty], which is especially useful on forms. You can apply this attribute to a property on a Razor Page (with or without an explicit PageModel) to opt into data binding for non-GET requests to the page. This enables tag helpers like asp-for and asp-validation-for to work with the property you've specified, and allows handlers to work with bound properties without having to specify them as method parameters. The [BindProperty] attribute also works on Controllers.

Figure 8 shows a Razor Page that lets users add new records to the application.

> Razor Pages can support strongly typed page models.

It's pretty common to have a page that supports more than one operation using the same HTTP verb. For example, the main page in the sample supports listing the entities (as the default GET behavior), as well as the ability to delete an entry or add a new entry (both as POST requests). Razor Pages support this scenario using named handlers, shown in **Figure 9**, which include the name after the verb (but before the "Async" suffix, if present). The PageModel

Figure 5 **Using Inline Variables**

```
@page
@{
  var swords = new List<string>()
  {
    "Katana",
    "Ninjago"
  };
}
<h2>Ninja Swords</h2>
<ul>
  @foreach (var item in swords)
  {
    <li>@item</li>
  }
</ul>
<a asp-page="/Ninjas/Index">Ninja List</a>
```

Figure 6 **Defining the Model**

```
@page
@using WithRazorPages.Core.Interfaces;
@using WithRazorPages.Core.Model;
@model IndexModel
@functions
{
  public class IndexModel : PageModel
  {
    private readonly IRepository<Zombie> _zombieRepository;

    public IndexModel(IRepository<Zombie> zombieRepository)
    {
      _zombieRepository = zombieRepository;
    }
    // additional code omitted
  }
}
```

base type is similar to the base Controller type in that it provides a number of helper methods you can use when returning action results. When performing updates like adding a new record, you often want to redirect the user immediately after the operation, if successful. This eliminates the issue of browser refreshes triggering duplicate calls to the server, resulting in duplicate records (or worse). You can use RedirectToPage with no arguments to redirect to the default GET handler of the current Razor Page.

You can specify a named handler using the asp-page-handler tag helper, applied to a form, link or button:

```
<a asp-page-handler="Handler">Link Text</a>
<button type="submit" asp-page-handler="delete" asp-route-id="@id">Delete</button>
```

The asp-page-handler tag uses routing to build the URL. The handler name and any asp-route-parameter attributes are applied as querystring values by default. The Delete button in the previous code generates a URL like this one:

```
Ninjas?handler=delete&id=1
```

If you'd rather have the handler as part of the URL, you can specify this behavior with the @page directive:

```
@page "{handler?}/{id?}"
```

With this route specified, the generated link for the Delete button would be:

```
Ninjas/Delete/1
```

Filters

Filters are another powerful feature of ASP.NET Core MVC (and one I covered in the August 2016 issue: msdn.microsoft.com/mt767699). If you're using a Page Model in a separate file, you can use attribute-based filters with Razor Pages, including placing filter attributes on the page model class. Otherwise, you can still specify global filters when you configure MVC for your app. One of the most common uses of filters is to specify authorization policies within your app. You can configure folder- and page-based authorization policies globally:

```
services.AddMvc()
  .AddRazorPagesOptions(options =>
  {
    options.Conventions.AuthorizeFolder("/Account/Manage");
    options.Conventions.AuthorizePage("/Account/Logout");
    options.Conventions.AllowAnonymousToPage("/Account/Login");
  });
```

Figure 7 **Details.cshtml—Displaying Details for a Given Record Id**

```
@page "{id:int}"
@using WithRazorPages.Core.Interfaces;
@using WithRazorPages.Core.Model;
@inject IRepository<Ninja> _repository

@functions {
  public Ninja Ninja { get; set; }

  public IActionResult OnGet(int id)
  {
    Ninja = _repository.GetById(id);

    // A void handler (no return) is equivalent to return Page()
    return Page();
  }
}
<h2>Ninja: @Ninja.Name</h2>
<div>
    Id: @Ninja.Id
</div>
<div>
    <a asp-page="..">Ninja List</a>
</div>
```

You can use all of the existing kinds of filters with Razor Pages except for Action filters, which apply only to action methods within Controllers. Razor Pages also introduce the new Page filter, represented by IPageFilter (or IAsyncPageFilter). This filter lets you add code that will run after a particular page handler has been selected, or before or after a handler method executes. The first method can be used to change which handler is used to handle a request, for example:

```
public void OnPageHandlerSelected(PageHandlerSelectedContext context)
{
  context.HandlerMethod =
    context.ActionDescriptor.HandlerMethods.First(m => m.Name == "Add");
}
```

Figure 8 **New.cshtml—Adds a New Plant**

```
@page
@using WithRazorPages.Core.Interfaces;
@using WithRazorPages.Core.Model;
@inject IRepository<Plant> _repository

@functions {
  [BindProperty]
  public Plant Plant { get; set; }

  public IActionResult OnPost()
  {
    if(!ModelState.IsValid) return Page();

    _repository.Add(Plant);

    return RedirectToPage("./Index");
  }
}
<h1>New Plant</h1>
<form method="post" class="form-horizontal">
  <div asp-validation-summary="All" class="text-danger"></div>
  <div class="form-group">
    <label asp-for="Plant.Name" class="col-md-2 control-label"></label>
    <div class="col-md-10">
      <input asp-for="Plant.Name" class="form-control" />
      <span asp-validation-for="Plant.Name" class="text-danger"></span>
    </div>
  </div>
  <div class="form-group">
    <div class="col-md-offset-2 col-md-10">
      <button type="submit" class="btn btn-primary">Save</button>
    </div>
  </div>
</form>
<div>
  <a asp-page="./Index">Plant List</a>
</div>
```

Figure 9 **Named Handlers**

```
public async Task OnGetAsync()
{
  Ninjas = _ninjaRepository.List()
    .Select(n => new NinjaViewModel { Id = n.Id, Name = n.Name }).ToList();
}

public async Task<IActionResult> OnPostAddAsync()
{
  var entity = new Ninja()
  {
    Name = "Random Ninja"
  };
_  ninjaRepository.Add(entity);

  return RedirectToPage();
}

public async Task<IActionResult> OnPostDeleteAsync(int id)
{
  var entityToDelete = _ninjaRepository.GetById(id);
_ ninjaRepository.Delete(entityToDelete);

  return RedirectToPage();
}
```

After a handler has been selected, model binding occurs. After model binding, the OnPageHandlerExecuting method of any page filters is called. This method can access and manipulate any model-bound data available to the handler, and can short circuit the call to the handler. The OnPageHandlerExecuted method is then called after the handler has executed, but before the action result executes.

Conceptually, page filters are very similar to action filters, which run before and after actions execute.

Note that one filter, ValidateAntiforgeryToken, isn't required for Razor Pages at all. This filter is used to protect against Cross-Site Request Forgery (CSRF or XSRF) attacks, but this protection is built into Razor Pages automatically.

Architectural Pattern

Razor Pages ship as part of ASP.NET Core MVC, and take advantage of many built-in ASP.NET Core MVC features like routing, model binding and filters. They share some naming similarity with the Web Pages feature that Microsoft shipped with Web Matrix in 2010. However, while Web Pages primarily targeted novice Web developers (and were of little interest to most experienced developers), Razor Pages combine strong architectural design with approachability.

Architecturally, Razor Pages don't follow the Model-View-Controller (MVC) pattern, because they lack Controllers. Rather, Razor Pages follow more of a Model-View-ViewModel (MVVM) pattern that should be familiar to many native app developers. You can also consider Razor Pages to be an example of the Page Controller pattern, which Martin Fowler describes as "An object that handles a request for a specific page or action on a Web site. That [object] may be the page itself, or it may be a separate object that corresponds to that page." Of course, the Page Controller pattern should also be familiar to anyone who has worked with ASP.NET Web Forms, because this was how the original ASP.NET pages worked, as well.

Unlike ASP.NET Web Forms, Razor Pages are built on ASP.NET Core and support loose coupling, separation of concerns and SOLID principles. Razor Pages are easily unit tested (if separate PageModel classes are used) and can provide a foundation for clean, maintainable enterprise applications. Don't write off Razor Pages as just a "training wheels" feature meant for hobbyist programmers. Give Razor Pages a serious look and consider whether Razor Pages (alone or in combination with traditional Controller and View pages) can improve the design of your ASP.NET Core application by reducing the number of folders you need to jump between when working on a particular feature.

Migrating

Although Razor Pages don't follow the MVC pattern, they're so closely compatible with the existing ASP.NET Core MVC Controllers and Views that switching between one and the other is usually very simple. To migrate existing Controller/View-based pages to use Razor Pages, follow these steps:

1. Copy the Razor View file to the appropriate location in the /Pages folder.
2. Add the @page directive to the View. If this was a GET-only View, you're done.
3. Add a PageModel file named viewname.cshtml.cs and place it in the folder with the Razor Page.
4. If the View had a ViewModel, copy it to a PageModel file.
5. Copy any actions associated with the view from its Controller to the PageModel class.
6. Rename the actions to use the Razor Pages handler syntax (for example, "OnGet").
7. Replace references to View helper methods with Page methods.
8. Copy any constructor dependency injection code from the Controller to the PageModel.
9. Replace code-passing model to views with a [BindProperty] property on the PageModel.
10. Replace action method parameters accepting view model objects with a [BindProperty] property, as well.

A well-factored MVC app will often have separate files for views, controllers, viewmodels, and binding models, usually each in separate folders in the project. Razor Pages allow you to consolidate these concepts into a couple of linked files, in a single folder, while still allowing your code to follow logical separation of concerns.

Razor Pages ship as part of ASP.NET Core MVC, and take advantage of many built-in ASP.NET Core MVC features like routing, model binding, and filters.

You should be able to reverse these steps to move from a Razor Pages implementation to a Controller/View-based approach, in most cases. Following these steps should work for most simple MVC-based actions and views. More complex applications may require additional steps and troubleshooting.

Next Steps

The sample includes four versions of the NinjaPiratePlantZombie organizer application, with support for adding and viewing each data type. The sample shows how to organize an app with several distinct functional areas using traditional MVC, MVC with Areas, MVC with Feature Slices and Razor Pages. Explore these different approaches and see which ones will work best in your own ASP.NET Core applications. The updated source code for this sample is available at bit.ly/2eJ01cS. ■

Steve Smith *is an independent trainer, mentor and consultant. He is a 14-time Microsoft MVP award recipient, and works closely with several Microsoft product teams. Contact him at ardalis.com or on Twitter: @ardalis if your team is considering a move to ASP.NET Core or if you're looking to adopt better coding practices.*

Thanks to the following Microsoft technical expert for reviewing this article: *Ryan Nowak*

NEW: HANDS-ON LABS

Join us for full-day, pre-conference hands-on labs Sunday, November 12.

Only $595 through Sept. 15

Whether you are an

- Engineer
- Developer
- Programmer
- Software Architect
- Software Designer

You will walk away from this event having expanded your .NET skills and the ability to build better applications.

REGISTER NOW

REGISTER BY SEPTEMBER 15 & SAVE $400!*

Use promo code ORLSEP4

*Savings based on 5-day packages only. See website for details.

LIVE! 360

TECH EVENTS WITH PERSPECTIVE

5 GREAT CONFERENCES, 1 GREAT PRICE

Visual Studio Live! Orlando is part of Live! 360, the Ultimate Education Destination. This means you'll have access to four (4) other co-located events at no additional cost:

- Visual Studio LIVE! — EXPERT SOLUTIONS FOR .NET DEVELOPERS
- SQL Server LIVE! — TRAINING FOR DBAs AND IT PROS
- TECHMENTOR — IN-DEPTH TRAINING FOR IT PROS
- Office & SharePoint LIVE! — ON-PREMISE, CLOUD & CROSS-PLATFORM TRAINING
- Modern Apps LIVE! — MOBILE, CROSS-DEVICE & CLOUD DEVELOPMENT

Five (5) events and hundreds of sessions to choose from—mix and match sessions to create your own, custom event line-up—it's like no other conference available today!

TURN THE PAGE FOR MORE EVENT DETAILS →

VSLIVE.COM/ORLANDOMSDN

PRODUCED BY

VIRTUALIZATION & Cloud Review | Visual Studio MAGAZINE | 1105 MEDIA YOUR GROWTH. OUR BUSINESS.

Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS

AGENDA AT-A-GLANCE

ALM / DEVOPS | CLOUD COMPUTING | NATIVE CLIENT | SOFTWARE PRACTICES | VISUAL STUDIO / .NET FRAMEWORK | WEB CLIENT

START TIME	END TIME	NEW Full Day Hands-On Labs: Sunday, November 12, 2017	
9:00 AM	6:00 PM	**VSS01** Full Day Hands-On Lab: Busy Developer's HOL on Angular - *Ted Neward*	**VSS02** Full Day Hands-On Lab: From 0-60 in a day with

START TIME	END TIME	Pre-Conference Workshops: Monday, November 13, 2017		
8:30 AM	5:30 PM	**VSM01** Workshop: Distributed Cross-Platform Application Architecture - *Jason Bock & Rockford Lhotka*	**VSM02** Workshop: Artificial Intelligence - *Brian Randell*	**VSM03** Workshop: Designing, Developing,
6:30 PM	8:00 PM	Dine-A-Round Dinner @ Universal CityWalk - 6:30pm - *Meet at Conference Registration Desk to walk over with the group*		

START TIME	END TIME	Day 1: Tuesday, November 14, 2017		
8:00 AM	9:00 AM	Visual Studio Live! KEYNOTE: To Be Announced		
9:15 AM	10:30 AM	**VST01** Front-end Web Development in 2017 for the Front-endally Challenged Microsoft Developer - *Chris Klug*	**VST02** Go Mobile with C#, Visual Studio, and Xamarin - *James Montemagno*	**VST03** Microservices with Azure Container Service & Service Fabric - *Vishwas Lele*
10:30 AM	11:00 AM	Networking Break • Visit the EXPO - *Pacifica 7*		
11:00 AM	12:00 PM	**LIVE! 360 KEYNOTE: To Be Announced** - *Pacifica 6*		
12:00 PM	12:45 PM	Lunch • Visit the EXPO		
12:45 PM	1:30 PM	Dessert Break • Visit the EXPO		
1:30 PM	2:45 PM	**VST05** ASP.NET Core MVC—What You Need to Know - *Philip Japikse*	**VST06** Optimizing and Extending Xamarin.Forms Mobile Apps - *James Montemagno*	**VST07** Tactical DevOps with Visual Studio Team Services - *Brian Randell*
2:45 PM	3:15 PM	Networking Break • Visit the EXPO - *Pacifica 7*		
3:15 PM	4:30 PM	**VST09** Angular(2)—The 75-Minute Crash Course - *Chris Klug*	**VST10** Building a Cross-Platform Mobile App Backend in the Cloud - *Nick Landry*	**VST11** Database Lifecycle Management and the SQL Server Database - *Brian Randell*
4:40 PM	5:00 PM	**VST13** Fast Focus: Aurelia vs. Just Angular - *Chris Klug*	**VST14** Fast Focus: Tips & Tricks for Xamarin Development - *James Montemagno*	**VST15** Fast Focus on Azure Functions
5:10 PM	5:30 PM	**VST17** Fast Focus: Web Security 101 - *Brock Allen*	**VST18** Fast Focus: Cross-Platform Code Reuse - *Rockford Lhotka*	**VST19** Fast Focus: Exploring Microservices in a Microsoft Landscape - *Marcel de Vries*
5:30 PM	7:30 PM	Exhibitor Reception - *Pacifica 7*		

START TIME	END TIME	Day 2: Wednesday, November 15, 2017		
8:00 AM	9:15 AM	**VSW01** User Authentication for ASP.NET Core MVC Applications - *Brock Allen*	**VSW02** Cloud Oriented Programming - *Vishwas Lele*	**VSW03** Overcoming the Challenges of Mobile Development in the Enterprise - *Roy Cornelissen*
9:30 AM	10:45 AM	**VSW05** Building AngularJS Component-Based Applications - *Miguel Castro*	**VSW06** Building Modern Web Apps with Azure - *Eric D. Boyd*	**VSW07** Creating a Release Pipeline with Team Services - *Esteban Garcia*
10:45 AM	11:30 AM	Networking Break • Visit the EXPO - *Pacifica 7*		
11:30 AM	12:30 PM	**LIVE! 360 KEYNOTE: To Be Announced** - *Pacifica 6*		
12:30 PM	1:30 PM	Birds-of-a-Feather Lunch		
1:30 PM	2:00 PM	Dessert Break • Visit the EXPO		
2:00 PM	3:15 PM	**VSW09** Securing Web APIs in ASP.NET Core - *Brock Allen*	**VSW10** A/B Testing, Canary Releases and Dark Launching, Implementing Continuous Delivery on Azure - *Marcel de Vries*	**VSW11** The Zen of UI Automation Testing
3:15 PM	4:00 PM	Networking Break • Visit the EXPO • Expo Raffle @ 3:30 p.m. - *Pacifica 7*		
4:00 PM	5:15 PM	**VSW13** Build Object-Oriented Enterprise Apps in JavaScript with TypeScript	**VSW14** Lock the Doors, Secure the Valuables, and Set the Alarm - *Eric D. Boyd*	**VSW15** Unit Testing & Test-Driven Development (TDD) for Mere Mortals - *Benjamin Day*
8:00 PM	10:00 PM	Live! 360 Dessert Luau - *Wantilan Pavilion*		

START TIME	END TIME	Day 3: Thursday, November 16, 2017		
8:00 AM	9:15 AM	**VSH01** HTTP/2: What You Need to Know - *Robert Boedigheimer*	**VSH02** PowerApps and Flow Part II: Package, Embed, and Extend Your Applications - *Manas Maheshwari & Pratap Ladhani*	**VSH03** Exploring C# 7 New Features - *Adam Tuliper*
9:30 AM	10:45 AM	**VSH05** ASP.NET Tag Helpers - *Robert Boedigheimer*	**VSH06** Storyboarding 101 - *Billy Hollis*	**VSH07** .NET Standard—From Noob to Ninja - *Adam Tuliper*
11:00 AM	12:00 PM	Visual Studio Live! Panel: To Be Announced - *Brian Randell (Moderator), Damian Brady, Jeremy Clark, Esteban Garcia, Billy Hollis, & Adam Tuliper*		
12:00 PM	1:00 PM	Lunch on the Lanai - *Lanai / Pacifica 7*		
1:00 PM	2:15 PM	**VSH09** I See You: Watching the User with Reactive Forms - *Deborah Kurata*	**VSH10** Continuous Integration and Deployment for Mobile Using Azure Services - *Kevin Ford*	**VSH11** Deploying Straight to Production: A Guide to the Holy Grail - *Damian Brady*
2:30 PM	3:45 PM	**VSH13** Angular Routing - *Deborah Kurata*	**VSH14** XAML Inception—Deep Composition for Better UI - *Billy Hollis*	**VSH15** Application Insights: Measure Your Way to Success - *Esteban Garcia*
4:00 PM	5:00 PM	Next? Visual Studio Live! Networking Event - *Brian Randell (Moderator), Damian Brady, Jeremy Clark, Esteban Garcia, Billy Hollis, & Deborah Kurata*		

START TIME	END TIME	Post-Conference Workshops: Friday, November 17, 2017	
8:00 AM	5:00 PM	**VSF01** Workshop: Angular Fundamentals - *John Papa*	**VSF02** Workshop: Building, Running & Microservices with Docker Containers on Azure

Speakers and sessions subject to change

Presented in Partnership with Magenic

Modern Apps LIVE!

MOBILE, CROSS-DEVICE & CLOUD DEVELOPMENT

LIVE! 360

TECH EVENTS WITH PERSPECTIVE

ROYAL PACIFIC RESORT AT UNIVERSAL ORLANDO

NOVEMBER 12-17

WEB SERVER	MODERN APPS LIVE!
Xamarin and Xamarin.Forms - *Roy Cornelissen*	
	Pre-Con Workshops: Monday, Nov. 13
Service Oriented Technologies: & Implementing WCF and the Web API - *Miguel Castro*	**MAM01** Workshop: Building Modern Mobile Apps - *Brent Edwards & Kevin Ford*
	Dine-A-Round Dinner
	Day 1: Tuesday, November 14, 2017
	Modern Apps Live! KEYNOTE PANEL: Industry Trends, Technology, and Your Career - *Matt Lockhart (Moderator)*
VST04 What's New in Visual Studio 2017 - *Robert Green*	**MAT01** Modern App Development: Transform How You Build Web and Mobile Software - *Rockford Lhotka*
	Networking Break • Visit the EXPO - *Pacifica 7*
	LIVE! 360 KEYNOTE
	Lunch • Visit the EXPO
	Dessert Break • Visit the EXPO
VST08 To Be Announced	**MAT02** Architecture: The Key to Modern App Success - *Brent Edwards*
	Networking Break • Visit the EXPO - *Pacifica 7*
VST12 Top 10 Entity Framework Core Features Every Developer Should Know - *Philip Japikse*	**MAT03** Focus on the User Experience #FTW - *Jim Barrett*
VST16 Fast Focus: Busting .NET Myths - *Jason Bock*	**MAT04** Fast Focus: Hybrid Web Frameworks - *Allen Conway*
VST20 Fast Focus: Dependency Injection in 20 Minutes - *Miguel Castro*	**MAT05** Fast Focus: Web Assembly - *Jason Bock*
	Exhibitor Reception - *Pacifica 7*
	Day 2: Wednesday, November 15, 2017
VSW04 Building Apps with Microsoft Graph and Visual Studio - *Robert Green*	**MAW01** Manage Distributed Teams with Visual Studio Team Services and Git - *Brian Randell*
VSW08 Bots are the New Apps: Building Bots with ASP.NET Web API & Language Understanding - *Nick Landry*	**MAW02** DevOps, Continuous Integration, the Cloud, and Docker - *Dan Nordquist*
	Networking Break • Visit the EXPO - *Pacifica 7*
	LIVE! 360 KEYNOTE
	Birds-of-a-Feather Lunch
	Dessert Break • Visit the EXPO
VSW12 To Be Announced	**MAW03** Security with Speed for Modern Developers - *Michael Lester*
	Networking Break • Visit the EXPO • Raffle @ 3:30 p.m.
VSW16 PowerApps, Flow, and Common Data Service: Empowering Businesses with the Microsoft Business Application Platform - *Charles Sterling*	**MAW04** Coding for Quality and Maintainability - *Jason Bock*
	Live! 360 Dessert Luau - *Wantilan Pavilion*
	Day 3: Thursday, November 16, 2017
VSH04 Top 10 Ways to Go from Good to Great Scrum Master - *Benjamin Day*	**MAH01** Modern Web Development: Building Server Side Using ASP.NET Core, MVC, Web API, and Azure - *Allen Conway*
VSH08 Devs vs. Ops: Making Friends with the Enemy - *Damian Brady*	**MAH02** Modern Web Development: Building Client Side Using TypeScript and Angular - *Allen Conway*
	Modern Apps Live! Panel: Mobile Development Technologies - *Rockford Lhotka (Moderator), James Montemagno, Kevin Ford*
	Lunch on the Lanai - *Lanai / Pacifica 7*
VSH12 Design Patterns: Not Just for Architects - *Jeremy Clark*	**MAH03** Using All That Data: Power BI to the Rescue - *Scott Diehl*
VSH16 DI Why? Getting a Grip on Dependency Injection - *Jeremy Clark*	**MAH04** Modern Mobile Development: Build a Single App For iOS, Android, and Windows with Xamarin Forms - *Kevin Ford*
	Next? Modern Apps Live! Networking Event - *Rockford Lhotka (Moderator)*
	Post-Con Workshops: Friday, Nov. 17
Continuously Deploying - *Marcel de Vries & Rene van Osnabrugge*	**MAF01** Workshop: Modern App Deep Dive—Xamarin, Responsive Web, UWP - *Kevin Ford, Brent Edwards, Allen Conway*

Check Out These Additional Sessions for Developers at Live! 360

Office & SharePoint LIVE!

ON-PREMISE, CLOUD & CROSS-PLATFORM TRAINING

Office & SharePoint Live! features 15+ developer sessions, including:

- **NEW!** Full Day Hands-On Lab: Developing Extensions for Microsoft Teams - *Paul Schaeflein*
- Workshop: Mastering the SharePoint Framework- *Andrew Connell*
- TypeScript for SharePoint Developers -*Rob Windsor*
- Building Office Add-ins for Outlook with Angular - *Andrew Connell*
- Developing SharePoint Framework Components Using Visual Studio - *Paul Schaeflein*
- What Every Developer Needs to Know about SharePoint Development Online or On-Prem - *Robert Bogue*
- Build a Complete Business Solution Using Microsoft Graph API through Client Side Web Parts - *Julie Turner*

SQL Server LIVE!

TRAINING FOR DBAs AND IT PROS

SQL Server Live! features 30+ developer sessions, including:

- **NEW!** Full Day Hands-On Lab: Developer Dive into SQL Server 2016 - *Leonard Lobel*
- Turbo Boost - SQL Tricks Everybody MUST Know - *Pinal Dave*
- Advanced SSIS Package Authoring with Biml - *Tim Mitchell*
- Graph DB Support in SQL Server 2017 - *Karen Lopez*
- Big Data Technologies: What, Where and How to Run Them on Azure - *Andrew Brust*
- Top Five SQL Server Query Tuning Tips - *Janis Griffin*
- Workshop: Big Data, BI, and Analytics on The Microsoft Stack - *Andrew Brust*

TECHMENTOR

IN-DEPTH TRAINING FOR IT PROS

TechMentor features 20 + developer sessions, including:

- **NEW!** Full Day Hands-On Lab: Ethical Hacking with Kali Linux - *Mike Danseglio & Avril Salter*
- Workshop: Windows Security—How I Do It! - *Sami Laiho*
- Hardware, Camtasia, and a Storyline: Creating Your Own User Training - *Greg Shields*
- Make Your PowerShell Scripts Bulletproof with Pester - *Melissa Januszko*
- Controlling Your Azure Spend - *Timothy Warner*
- PowerShell Scripting Secrets - *Jeffery Hicks*
- In-Depth Introduction to Docker - *Neil Peterson*

VSLIVE.COM/ORLANDOMSDN

Snapshot Debugging for Production Apps and Services in Azure

Nikhil Joglekar

The mantra of DevOps is to "move fast and break things." However, this requires you to constantly fix the things that break. In production environments, detecting and diagnosing an issue can be costly in terms of time and resources. Issues that only manifest in production problems are among the hardest to solve. How often have you run into a problem in production that worked just fine on your development box or had to sift through thousands of logs to try to understand what caused the issue?

Currently, there are several ways you can investigate reported site issues:

Try to reproduce the issue locally. A production environment is a lot more complex than a development environment: It's often on a larger scale and houses real data in the production database. Getting a local reproduction of the issue might involve copying production data into a staging or testing environment. In the worst-case scenario, it might not be possible to reproduce an issue locally.

This article discusses:

- Debugging live production services in Azure App Services safely
- Using Snappoints to debug apps without affecting users
- Using Logpoints to add additional logging to your app

Technologies discussed:

Visual Studio, Azure App Services, ASP.NET, Snapshot Debugger, Snappoints, Logpoints

Sift through logs to try to find the cause of the issue. Logs will ideally contain details and context around the issue at the exact moment it happens. There's ample tooling for collecting, aggregating and searching through logs, yet trying to find the exact details to correlate with the reported issue can be like looking for a needle in a haystack. Often, your app logs won't contain the details necessary to identify the root cause of the problem. In such cases, you'll have to add additional logs to your app and redeploy to get the details you need.

Call up the customer for more context. If logs and local reproduction prove fruitless, you might have to get directly in touch with the customer who ran into the issue. The customer might have interacted with the app in an unexpected or untested way.

Attach a real debugger as a last-ditch effort. Attaching a debugger into a production environment will let you inspect the state of the app, but setting and hitting a breakpoint will effectively stop the app from serving requests while you're debugging.

These methods each have their pros and cons, yet even the most experienced developers following the loop can find diagnosing production issues time-consuming and difficult. In this article, I'll introduce the Snapshot Debugger for Azure, a set of new tools in Visual Studio that can drastically reduce the time and pain in dealing with production issues.

Snapshot Debugger was designed from the ground up as a tool safe to debug production services running in Azure. It lets you debug live site issues with almost no impact to the service. Snapshot Debugging brings the ease of using a debugger to a cloud scale. There are three key functionalities in Snapshot Debugger:

Snappoints. Like a breakpoint you set in your app while debugging on your development box, Snappoints are placed on a line of code you specify. When your app hits a Snappoint, the Snapshot Debugger takes a snapshot of the app, letting you inspect the call stack, local variables and objects on the heap. Unlike breakpoints, the app doesn't stop. Customers interacting with your app won't realize that you're currently debugging your app.

Logpoints. These let you insert additional log statements into a live production service in cases where you might not have log messages that detail a production issue. These new log statements are temporary and cause no side effects to your app.

Conditions. Only a fraction of the number of requests hitting a live app might be interesting or result in error. Conditional statements you specify apply to both Snappoints and Logpoints to ensure that you only get diagnostics information on interesting requests that might help identify the root cause of the problem.

Currently, the Snapshot Debugger is in public preview for ASP.NET and ASP.NET Core apps running on Azure App Services. You can download and try out the Snapshot Debugger at aka.ms/snappoint.

Debug Safely in Production with Snappoints

In the example in **Figure 1**, Visitors is an ASP.NET MVC Web site running on Azure App Services that registers and tracks visitors to Microsoft offices.

To set up this sample scenario: One of the registered visitors was seen exiting the Secret Project building, a building visitors should absolutely not have access to! You've frantically fired up your computer and started trying to figure out how the app gave the visitor access to the building. The first thing you did was look through the app logs for suspicious security access grants to any of the visitors. Unfortunately, you weren't able to find any incriminating logs.

In this app, visitors are represented by a "Visitor" object, which has an internal SecurityCode property that lets them access certain buildings. The only way visitors can enter a building is if the SecurityCode property associated with them has granted them the right level of access. This property is only handled internally by the app, but it's possible someone could somehow force in an unverified SecurityCode. To investigate this further, the version of the source code that's running in production in Visual Studio should be opened and the Snapshot Debugger started (this is done by right-clicking on Start Snapshot Debugger on an Azure App Service in the Cloud Explorer). Visual Studio will then connect to the production app running in Azure and enter a Snapshot Debugger session, as shown in **Figure 2**.

The Snapshot Debugger session functions a lot like a local debugging session on a development box. You can place Snappoints on a line you're interested in debugging, instead of placing breakpoints like you do locally. You place Snappoints in the same gutter that you place breakpoints in a local debug session, but they're inactive until you hit the Start/Update Collection button that turns Snappoints on in your production environment. Each Snappoint will only capture one snapshot by default: This snapshot reflects one request made to your app.

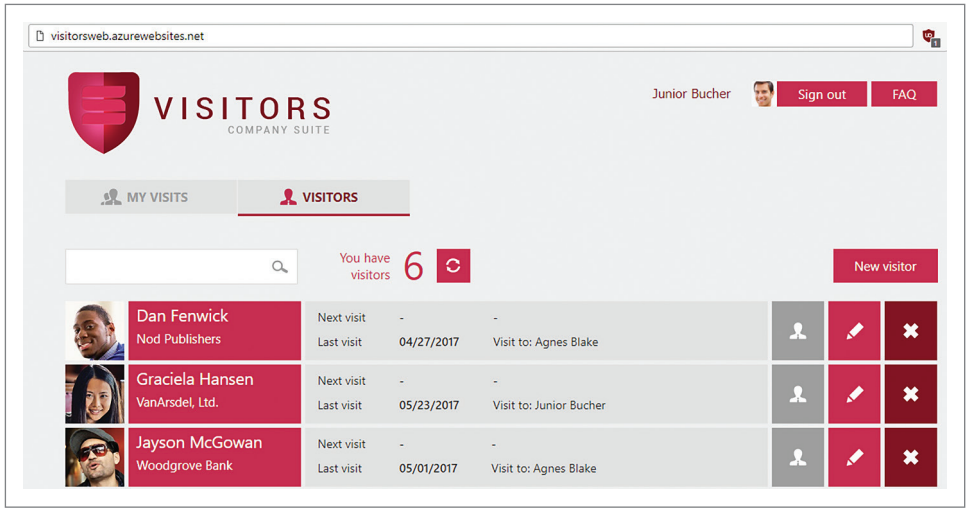


Figure 1 **Visitors Is an ASP.NET MVC Web Site Running on Azure App Services**

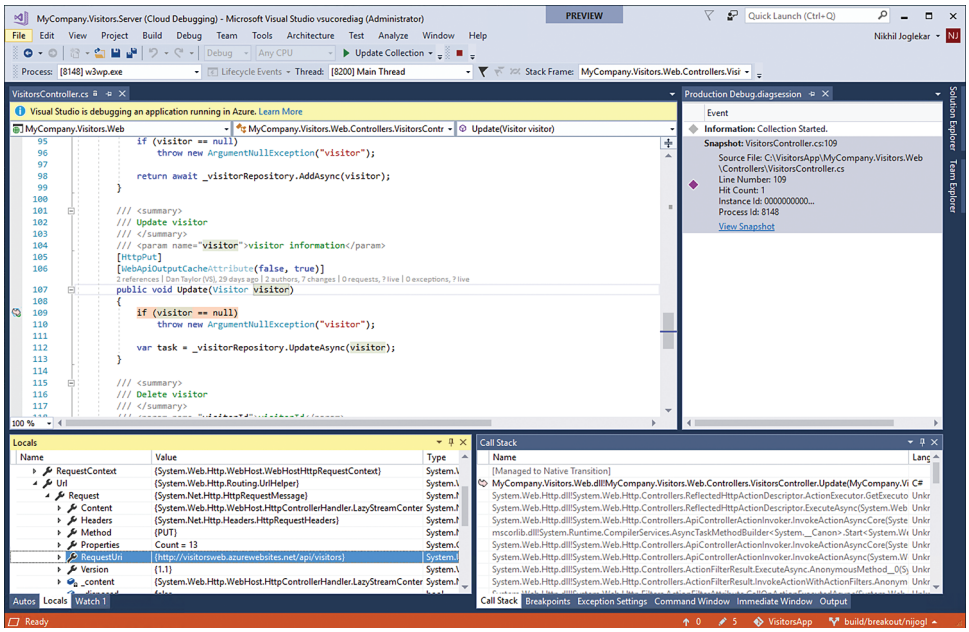


Figure 2 **A Snapshot Debugger Session**

```
107     public void Update(Visitor visitor)
108     {
109         if (visitor == null)
110             throw new ArgumentNullException("visitor");
111
112         var task = _visitorRepository.UpdateAsync(visitor);
113     }
114
```

Figure 3 **The Update Controller**

There are a few locations where the Visitors app updates visitors (and their SecurityCodes). The Update controller is one of those locations. You can investigate if there's an Update request that sets a SecurityCode by placing a Snappoint inside the Update method and turning it on with the Start Collection button, as shown in **Figure 3**.

> The Snapshot Debugger session functions a lot like a local debugging session on your development box.

The Snapshot Debugger takes a snapshot when the app in production runs a line with a Snappoint. This snapshot is captured in about 10ms, after which the app continues to execute requests. This snapshot will show up in Visual Studio inside the Diagnostic Tools Window on the right side; you can double-click it to open the snapshot. Opening the snapshot will give you all kinds of debug details as to what happened at the point of time when the line of code where you placed the Snappoint runs. You can view the call stack and Locals in their respective tool windows. You can go to the Watch window and add visitor.SecurityCode to inspect if there actually was a SecurityCode accepted by the app:

```
visitor.SecurityCode  null  string
```

SecurityCode on this snapshot is just as expected: null. The Snapshot Debugger captures a snapshot on the first request to hit the line of code where you placed a Snappoint. This snapshot reflects the correct behavior of your app and therefore doesn't help you diagnose the issue. A snapshot where the SecurityCode isn't null will help you investigate your hypothesis. To attempt to capture a snapshot when a SecurityCode is present, you can add conditions to the Snappoints by clicking the Options gear when hovering over a Snappoint, as shown in **Figure 4**.

The added conditional statement will let you narrow down and only take a snapshot on an interesting request. After turning on the new Snappoint by clicking Start/Update Collection and waiting for a few seconds, you see another snapshot visible in the Diagnostic Tools Window. Double-clicking to view the debug information for this snapshot shows the following information for the visitor local variable:

```
visitor  {MyCompany.Visitors.Model.Visitor}  MyCompany.Visitors.Model.Visitor
  Company  "Microsoft"  string
  Email  "toroidking@microsoft.com"  string
  FirstName  "Jackson"  string
  LastName  "Davis"  string
  Position  "Principal Software Engineer"  string
  SecurityCode  "SecretProjects"  string
  VisitorId  7  int
```

Bam! It looks as if your app somehow accepted an unverified SecurityCode. It appears as if someone had manually modified part of the HTTP request's payload into the Update controller to add in a SecurityCode. Your app then accepted the SecurityCode without any further checks. To fix this problem, you could either make the SecurityCode property private or add in checks for clients manually setting a SecurityCode.

The manual modification of the PUT request was not a scenario that you built your app to handle, nor had this scenario been tested locally. Snappoints let you investigate real customer requests that come into your app to detect the error with almost no impact to the site itself.

Get More Information On Demand with Logpoints

Now that you've identified the source of your error, you can fix your code and deploy it into production. However, it might be some time before you can make the fix, get it reviewed by your team and have it propagate through your continuous integration systems up to production. For now, you'll keep track of every user who tries to set a SecurityCode.

Logpoints let you add this additional temporary logging message to your app without any restart or redeployment and they let you evaluate variables to put in log messages. To add a Logpoint to your app, go to the settings on a Snappoint and convert it into a Logpoint (note any conditions you set will also work for Logpoints). Instead of taking a snapshot when the line of code runs, the app will emit a new log message. Logpoints will stay active only during a Snapshot Debugger session in Visual Studio. In the example in **Figure 5**, you log the visitor's name when an invalid SecurityCode is set.

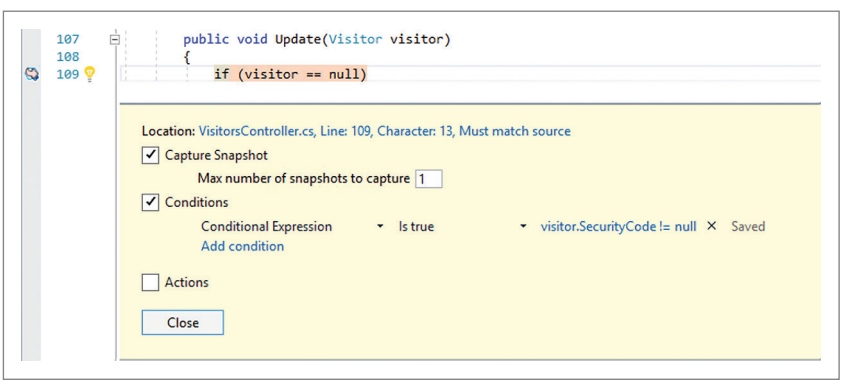


Figure 4 **Capturing a Snapshot When SecurityCode Is Present**

Logpoints can send logs to two places: back to Visual Studio in a live stream and to your app's log store. When Send to Output Window is checked, new log messages stream back into Visual Studio and display in both the Output Window and the Diagnostic Tools Window. When Send to app logs is checked, you send the logs back to your app's log store. To achieve this, you make a System.Diagnostics.Trace call with the input log message for ASP.NET apps and an ILogger.LogInformation call for ASP.NET Core

GROUPDOCS
Document Manipulation APIs

Manipulating Files?

APIs to view, export, annotate, compare, sign, automate and search documents in your applications.

GroupDocs.Total

GroupDocs.**Viewer**
GroupDocs.**Annotation**
GroupDocs.**Conversion**
GroupDocs.**Comparison**
GroupDocs.**Signature**
GroupDocs.**Assembly**
GroupDocs.**Metadata**
GroupDocs.**Search**
GroupDocs.**Text**

Try for Free

.NET Libraries

Java Libraries

Cloud APIs

Contact Us:
US: +1 903 306 1676
EU: +44 141 628 8900
AU: +61 2 8006 6987
sales@asposeptyltd.com

Visit us at www.groupdocs.com

GROUPDOCS
Document Manipulation APIs

apps. If you configure your logging framework to listen to System.Diagnostics or ILogger, the logs will appear in conjunction with any other app logs you currently collect.

Logpoints don't cause any side effects to your app in production. The Snapshot Debugger virtually executes the input log statements in a way that can't alter the state or execution of your app in production.

By using Logpoints, you're now tracking any user who's trying to use a SecurityCode in your app. You're collecting new log information in a part of your app that previously generated no logs. Logpoints let you get additional contextual information about your app, only where and when you need it.

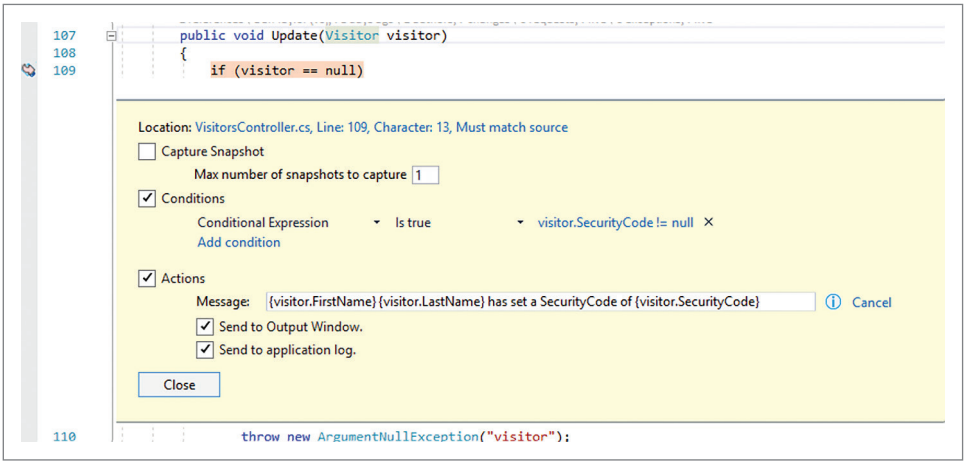


Figure 5 **Adding a Logpoint**

How Snapshot Debugging Is Production-Ready

The Snapshot Debugger aims to bring the ease of the Visual Studio debugging experience to a production scale for apps running in Azure. The process of capturing snapshots on a live app is minimally intrusive. Performance measurements show that Snapshot Debugger has a negligible impact to the speed of your app.

Snapshot Debugger is intended to be used on Release build versions of your app. Currently, debugging Release versions of managed apps can be challenging, as debugger controller might be slightly off due to function optimization and in-lining in Release build apps. The Snapshot Debugger gives you much better support for debugging Release build apps. It recompiles functions containing Snappoints with optimizations and in-lining temporarily disabled. Resulting snapshots in these functions are, therefore, more easily "debuggable." When you end your Snapshot Debugger session, the functions are recompiled once more and returned to their original production state.

When the Snapshot Debugger captures a snapshot of your app, it creates a fork of the app's process. This snapshot, or forked process, reflects the full state of the app and the app's heap at the point of time when the snapshot was taken. However, this snapshot doesn't immediately copy the full heap of the app; instead, it only copies the page table and sets pages to copy on write. If some of the app's objects on the heap change, their respective pages are then copied into the forked process. Each snapshot, therefore, requires only a small memory cost (on the Visitors app used in the example, the memory consumed by each snapshot was in the hundreds of kilobytes).

The Snapshot Debugger aggressively throttles the number of snapshots created to ensure that it puts minimal memory pressure on your production server. The Snapshot Debugger won't take snapshots if the amount of free memory on your server is too low. Additionally, snapshots only exist while the Snapshot Debugger session in Visual Studio is active. When you stop the Snapshot Debugger session, all snapshot processes are killed.

Snappoints also let you investigate a request as it progresses through different lines of your code. Unlike breakpoints, you can't step between Snappoints. However, you can place multiple Snappoints at different locations of interest in your code. The Start/Update Collection button will turn on all the Snappoints you place in bulk. Two Snappoints in one function will result in two snapshots in the Diagnostic Tools Window, assuming both lines with Snappoints execute. You can switch between the snapshots to see how variables change from the first snapshot to the second. The Snapshot Debugger will ensure that groups of Snappoints enabled in bulk will result in related snapshots. Two snapshots in one function will both be from the same request, even if there are hundreds of requests hitting your app at that moment in time.

In a production environment, there can be many servers running identical instances of your app. In some cases, interesting requests that might reflect an error may be rare and only occur on specific servers. The Snapshot Debugger enables you to investigate these issues as it supports debugging against multiple instances of your app at once. Snappoints placed in a function activate across every server running an instance of your app. Only the first instance to execute the line with the Snappoint will capture a snapshot. You can, therefore, use conditional statements to analyze seldom-occurring issues. The Snapshot Debugger will take the resulting snapshot only on the server where the input conditions become true.

Wrapping Up

The Snapshot Debugger was built after years of working with developers and listening to their difficulties in debugging and diagnosing production issues. The Snapshot Debugger enables you to have a rich diagnostics experience when developing .NET apps for Azure, in turn letting you save time and money when you run into inevitable production issues.

The Snapshot Debugger is currently in public preview for ASP.NET and ASP.NET Core apps running on Azure App Services. You can download the Snapshot Debugger and try it out at aka.ms/snappoint. ■

Nikhil Joglekar *is a program manager at Microsoft, focusing on debugging and diagnostics for Azure services. He has worked on the Snapshot Debugger, Visual Studio Profiler and Azure SDK since joining Microsoft two years ago. Contact him at Nikhil.Joglekar@microsoft.com or on Twitter: @nikjogo.*

Thanks to the following Microsoft technical experts for reviewing this article: *Jackson Davis and Andy Sterland*

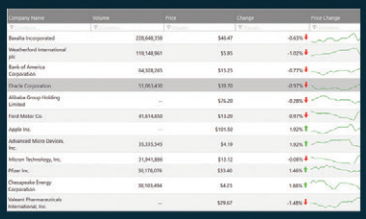
INFRAGISTICS

Ignite UI for JavaScript

Quickly create high performing JavaScript / HTML5 apps with 70+ ASP.NET MVC and native Angular UI Components

Build intuitive UI's for line-of-business applications with the industry's top performing grids and charts.

Download your free trial today at:
Infragistics.com/Ignite-ui



Fast, Lightweight Grids

Ignite UI delivers best-in-class performance for your high-demand data needs. The virtualized, load-on-demand data grids can handle any data scenario.

High Performance Charts

Choose one of many HTML5 chart types with statistical and technical indicators, trend lines, and more to deliver high performing and beautiful touch-friendly dashboards for desktop and mobile apps. Ideal for financial applications.

Build Modern Apps

Ignite UI plays nice with all your favorite app frameworks, data binding libraries and IDEs. With full support for AngularJS, Angular, KnockOut.js, ASP.NET MVC, Bootstrap, React and more.

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

JAMES MCCAFFREY

Deep Neural Network Training

A regular feed-forward neural network (FNN) has a set of input nodes, a set of hidden processing nodes and a set of output nodes. For example, if you wanted to predict the political leaning of a person (conservative, moderate, liberal) based on their age and income, you could create an FNN with two input nodes, eight hidden nodes and three output nodes. The number of input and output nodes is determined by the structure of your data, but the number of hidden nodes is a free parameter that you have to determine using trial and error.

A deep neural network (DNN) has two or more hidden layers. In this article I'll explain how to train a DNN using the back-propagation algorithm and describe the associated "vanishing gradient" problem. After reading this article, you'll have code to experiment with, and a better understanding of what goes on behind the scenes when you use a neural network library such as the Microsoft Cognitive Toolkit (CNTK) or Google TensorFlow.

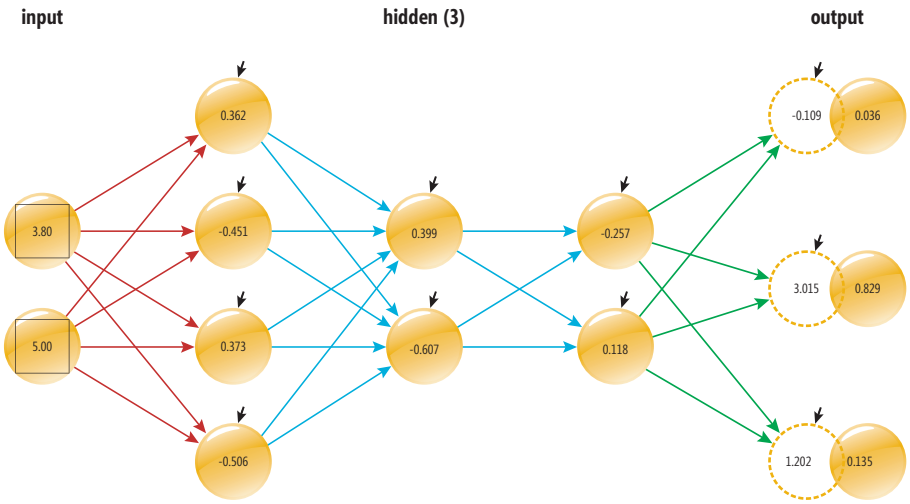


Figure 1 **An Example 2-(4,2,2)-3 Deep Neural Network**

Take a look at the DNN shown in **Figure 1**. The DNN has three hidden layers that have four, two and two nodes. The input node values are 3.80 and 5.00, which you can imagine as the normalized age (38 years old) and income ($50,000 per year) of a person. Each of the eight hidden nodes has a value between -1.0 and +1.0 because the DNN uses tanh activation.

The preliminary output node values are (-0.109, 3.015, 1.202). These values are converted to the final output values of (0.036, 0.829, 0.135) using the softmax function. The purpose of softmax is to coerce the output node values to sum to 1.0 so that they can be interpreted as probabilities. Assuming the output nodes represent the probabilities of "conservative", "moderate" and "liberal," respectively, the person in this example is predicted to be a political moderate.

In **Figure 1**, each of the arrows connecting nodes represents a numeric constant called a weight. Weight values are typically between -10.0 and +10.0, but can be any value in principle. Each of the small arrows pointing into the hidden nodes and the output nodes is a numeric constant called a bias. The values of the weights and the biases determine the output node values.

Code download available at msdn.com/magazine/0917magcode.

The process of finding the values of the weights and the biases is called training the network. The idea is to use a large set of training data, which has known input values and known correct output values, and then use an optimization algorithm to find values for the weights and biases so that the difference between computed output values and known correct output values is minimized. There are several algorithms that can be used to train a DNN, but by far the most common is the back-propagation algorithm.

This article assumes you have a basic understanding of the neural network input-output mechanism and at least intermediate level programming skills. The demo program is coded using C# but you shouldn't have too much trouble refactoring the demo to another language such as Python or Java if you wish. The demo program is too long to present in its entirety in this article, but the complete program is available in the accompanying code download.

The Demo Program

The demo begins by generating 2,000 synthetic training data items. Each item has four input values between -4.0 and +4.0, followed by three output values, which can be (1, 0, 0) or (0, 1, 0) or (0, 0, 1), representing three possible categorical values. The first and last training items are:

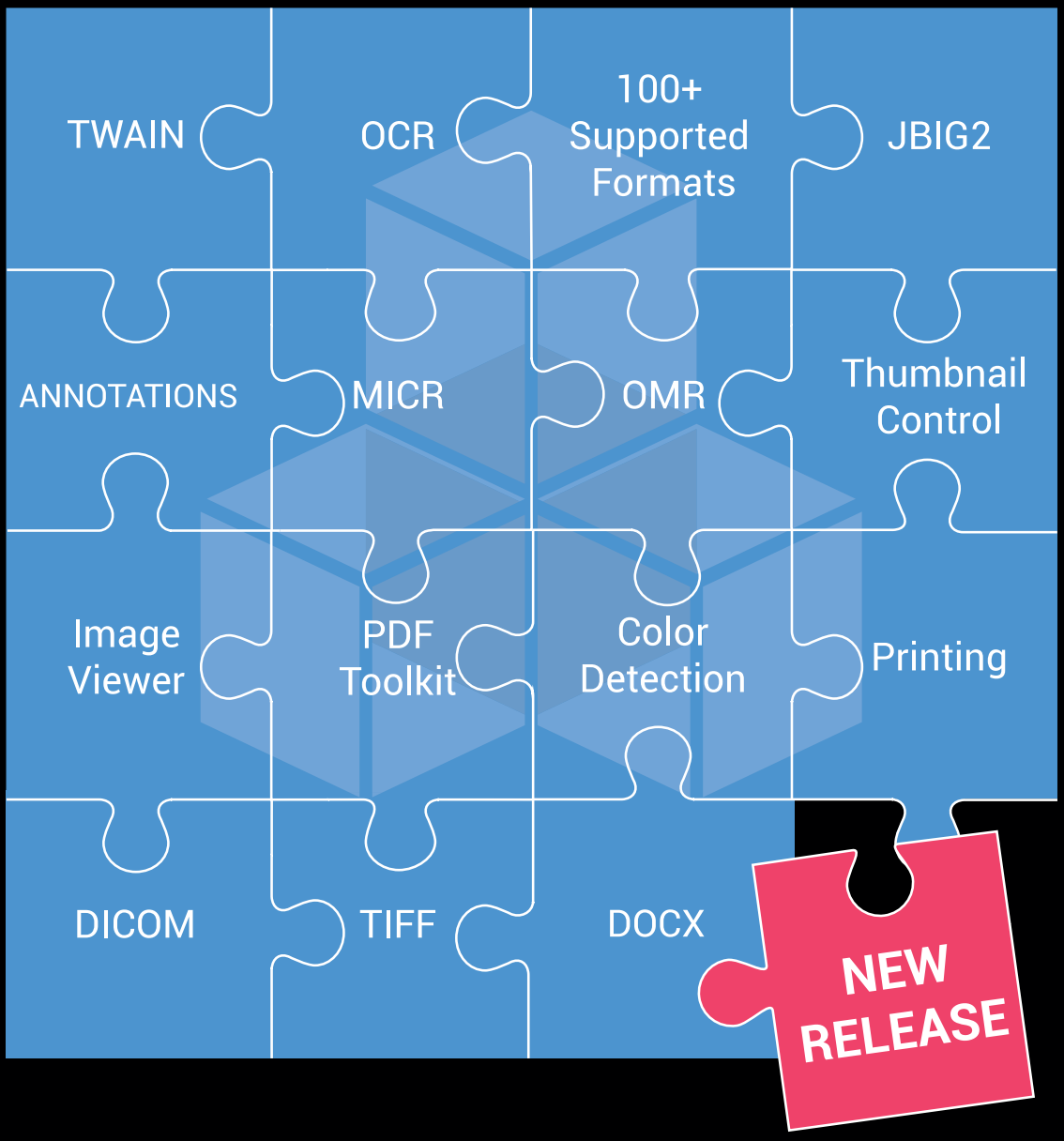
```
[   0]  2.24  1.91  2.52  2.41  0.00  1.00  0.00
...
[1999]  1.30 -2.41 -3.18  0.11  1.00  0.00  0.00
```

Behind the scenes, the dummy data is generated by creating a 4-(10,10,10)-3 DNN with random weights and biases values, and then feeding random input values to the network. After the

GdPicture.NET 14

100% ROYALTY FREE

Imaging SDK For **WinForms**, **WPF** And **Web** Development



Leverage your apps. with **GdPicture.NET** Imaging Toolkit

DOWNLOAD YOUR FREE TRIAL **www.gdpicture.com**

GdPicture.NET is an ORPALIS imaging technologies product

training data is generated, the demo creates a new 4-(10,10,10)-3 DNN and trains it using the back-propagation algorithm. During training, the current mean squared error and classification accuracy are displayed every 200 iterations.

The error slowly decreases and the accuracy slowly increases, as you'd expect. After training completes, the final accuracy of the DNN model is 93.45 percent, which means that 0.9345 * 2000 = 1869 items were correctly classified and therefore 131 items were incorrectly classified. The demo code that generates the output begins with:

```
using System;
namespace DeepNetTrain
{
  class DeepNetTrainProgram  {
    static void Main(string[] args) {
      Console.WriteLine("Begin deep net demo");
      int numInput = 4;
      int[] numHidden = new int[] { 10, 10, 10 };
      int numOutput = 3;
...
```

The demo program uses only plain C# with no namespaces except for System. First, the DNN to generate the simulated training data is prepared. The number of hidden layers, 3, is passed implicitly as the number of items in the numHidden array. An alternative design is to pass the number of hidden layers explicitly. Next, the training data is generated using helper method MakeData:

```
int numDataItems = 2000;
Console.WriteLine("Generating " + numDataItems +
  " artificial training data items ");
double[][] trainData = MakeData(numDataItems,
  numInput, numHidden, numOutput, 5);
Console.WriteLine("Done. Training data is: ");
ShowMatrix(trainData, 3, 2, true);
```

The 5 passed to MakeData is a seed value for a random object so that demo runs will be reproducible. The value of 5 was used only because it gave a nice demo. The call to helper ShowMatrix displays the first 3 rows and the last row of the generated data, with 2 decimal places, showing indices (true). Next, the DNN is created and training is prepared:

```
Console.WriteLine("Creating a 4-(10,10,10)-3 DNN");
DeepNet dn = new DeepNet(numInput, numHidden, numOutput);
int maxEpochs = 2000;
double learnRate = 0.001;
double momentum = 0.01;
```

The demo uses a program-defined DeepNet class. The back-propagation algorithm is iterative so a maximum number of iterations, 2,000 in this case, must be specified. The learning rate parameter controls how much the weights and bias values are adjusted each time a training item is processed. A small learning rate could result in training being too slow (hours, days or more) but a large learning rate could lead to wildly oscillating results that never stabilize. Picking a good learning rate is a matter of trial and error and is a major challenge when working with DNNs. The momentum factor is somewhat like an auxiliary learning rate, and typically speeds up training when a small learning rate is used.

The demo program calling code concludes with:

```
...
  double[] wts = dn.Train(trainData, maxEpochs,
    learnRate, momentum, 10);
  Console.WriteLine("Training complete");
  double trainError = dn.Error(trainData, false);
  double trainAcc = dn.Accuracy(trainData, false);
  Console.WriteLine("Final model MS error = " +
    trainError.ToString("F4"));
  Console.WriteLine("Final model accuracy = " +
    trainAcc.ToString("F4"));
  Console.WriteLine("End demo ");
}
```

The Train method uses the back-propagation algorithm to find values for the weights and biases so that the difference between computed output values and correct output values is minimized. The values of both the weights and biases are returned by Train. The argument of 10 passed to Train means to display progress messages every 2,000 / 10 = 200 iterations. It's important to monitor progress because bad things can, and often do, happen when training a neural network.

After training completes, the final error and accuracy of the model are calculated and displayed using the final weights and bias values, which are still inside the DNN. The weights and biases could have been explicitly reloaded by executing the statement dnn.SetWeights(wts), but it's not necessary in this case. The "false" arguments passed to methods Error and Accuracy mean to not display diagnostic messages.

Deep Neural Network Gradients and Weights

Each weight and bias in a DNN has an associated gradient value. A gradient is a calculus derivative of the error function and is just a value, such as -1.53, where the sign of the gradient tells you if the associated weight or bias should be increased or decreased to reduce error, and the magnitude of the gradient is proportional to how much the weight or bias should change. For example, suppose one of the weights, w, in a DNN has a value of +4.36, and after a training item is processed, the gradient for the weight, g, is calculated to be +2.50. If the learning rate, lr, is set to 0.10 then the new weight value is:

$$
\begin{aligned}
w &= w + (lr * g) \\
&= 4.36 + (0.10 * 2.50) \\
&= 4.36 + 0.25 \\
&= 4.61
\end{aligned}
$$

So, training a DNN really boils down to finding the gradients for each weight and bias value. As it turns out, calculating the gradients for the weights connecting the last hidden layer nodes to the output layer nodes, and the gradients for the output node biases is relatively easy even though the underlying math is extraordinarily profound. Expressed in code, the first step is to compute what's called the output node signals for each output node:

```
for (int k = 0; k < nOutput; ++k) {
  errorSignal = tValues[k] - oNodes[k];
  derivative = (1 - oNodes[k]) * oNodes[k];
  oSignals[k] = errorSignal * derivative;
}
```

Local variable errorSignal is the difference between the target value (the correct node value from the training data) and the computed output node value. The details can be very tricky. For example, the demo code uses (target - output) but some references use (output - target), which affects whether the associated weight update statement should add or subtract when modifying weights.

Local variable derivative is a calculus derivative (not the same as the gradient, which is also a derivative) of the output activation function, which in this case is the softmax function. In other words, if you use something other than softmax, you'll have to modify the calculation of the derivative local variable.

After the output node signals have been computed, they can be used to compute the gradients for the hidden-to-output weights:

Extreme Performance Linear Scalability

NCache™

Distributed Cache

- In-Memory App Data Caching
- ASP.NET Sessions & View State
- Runtime Data Sharing

NosDB

NoSQL Database

- Schema-Free JSON Documents
- Multiple Shards & Data Replication
- Powerful SQL, LINQ, ADO.NET

100% Native .NET

Open Source

sales@alachisoft.com

alachisoft.com

US: +1 (925) 236 3830

```
for (int j = 0; j < nHidden[lastLayer]; ++j) {
  for (int k = 0; k < nOutput; ++k) {
    hoGrads[j][k] = hNodes[lastLayer][j] * oSignals[k];
  }
}
```

In words, the gradient for a weight connecting a hidden node to an output node is the value of the hidden node times the output signal of the output node. After the gradient associated with a hidden-to-output weight has been computed, the weight can be updated:

```
for (int j = 0; j < nHidden[lastLayer]; ++j) {
  for (int k = 0; k < nOutput; ++k) {
    double delta = hoGrads[j][k] * learnRate;
    hoWeights[j][k] += delta;
    hoWeights[j][k] += hoPrevWeightsDelta[j][k] * momentum;
    hoPrevWeightsDelta[j][k] = delta;
  }
}
```

First the weight is incremented by delta, which is the value of the gradient times the learning rate. Then the weight is incremented by an additional amount—the product of the previous delta times the momentum factor. Note that using momentum is optional, but almost always done to increase training speed.

To recap, to update a hidden-to-output weight, you calculate an output node signal, which depends on the difference between target value and computed value, and the derivative of the output node activation function (usually softmax). Next, you use the output node signal and the hidden node value to compute the gradient. Then you use the gradient and the learning rate to compute a delta for the weight, and update the weight using the delta.

Unfortunately, calculating the gradients for the input-to-hidden weights and the hidden-to-hidden weights is much more complicated. A thorough explanation would take pages and pages, but you can get a good idea of the process by examining one part of the code:

```
int lastLayer = nLayers - 1;
for (int j = 0; j < nHidden[lastLayer]; ++j) {
  derivative = (1 + hNodes[lastLayer][j]) *
    (1 - hNodes[lastLayer][j]); // For tanh
  double sum = 0.0;
  for (int k = 0; k < nOutput; ++k) {
    sum += oSignals[k] * hoWeights[j][k];
  }
  hSignals[lastLayer][j] = derivative * sum;
}
```

This code calculates the signals for the last hidden layer nodes—those just before the output nodes. The local variable derivative is the calculus derivative of the hidden layer activation function, tanh in this case. But the hidden signals depend on a sum of products that involves the output node signals. This leads to the "vanishing gradient" problem.

The Vanishing Gradient Problem

When you use the back-propagation algorithm to train a DNN, during training the gradient values associated with hidden-to-hidden weights quickly become very small or even zero. If a gradient value is zero, then the gradient times the learning rate will be zero, and the weight delta will be zero, and the weight will not change. Even if a gradient doesn't go to zero, but gets very small, the delta will be tiny and training will slow to a crawl.

The reason gradients quickly head toward zero should be clear if you carefully examine the demo code. Because output node values are coerced to probabilities, they're all between 0 and 1. This leads to output node signals that are between 0 and 1. The multiplication part of computing the hidden node signals therefore involves repeatedly multiplying values between 0 and 1, which will result in smaller and smaller gradients. For example, 0.5 * 0.5 * 0.5 * 0.5 = 0.0625. Additionally, the tanh hidden layer activation function introduces another fraction-times-fraction term.

The demo program illustrates the vanishing gradient problem by spying on the gradient associated with the weight from node 0 in the input layer to node 0 in the first hidden layer. The gradient for that weight decreases quickly:

```
epoch = 200    gradient = -0.002536
epoch = 400    gradient = -0.000551
epoch = 600    gradient = -0.000141
epoch = 800    gradient = -0.159148
epoch = 1000  gradient = -0.000009
...
```

The gradient temporarily jumps up at epoch 800 because the demo updates weights and biases after every training item is processed (this is called "stochastic" or "online" training, as opposed to "batch" or "mini-batch" training), and by pure chance the training item processed at epoch 800 led to a larger than normal gradient.

> Long short-term memory networks are extremely good at natural language processing.

In the early days of DNNs, perhaps 25 to 30 years ago, the vanishing gradient problem was a show-stopper. As computing power increased, the vanishing gradient became less of a problem because training could afford to slow down a bit. But with the rise of very deep networks, with hundreds or even thousands of hidden layers, the problem resurfaced.

Many techniques have been developed to tackle the vanishing gradient problem. One approach is to use the rectified linear unit function (ReLU) instead of the tanh function for hidden layer activation. Another approach is to use different learning rates for different layers—larger rates for layers closer to the input layer. And the use of GPUs for deep learning is now the norm. A radical approach is to avoid back-propagation altogether, and instead use an optimization algorithm that doesn't require gradients, such as particle swarm optimization.

Wrapping Up

The term deep neural network most often refers to the type of network described in this article—a fully connected network with multiple hidden layers. But there are many other types of deep neural networks. Convolutional neural networks are very good at image classification. Long short-term memory networks are extremely good at natural language processing. Most of the variations of deep neural networks use some form of back-propagation and are subject to the vanishing gradient problem. ■

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

Thanks to the following Microsoft technical experts who reviewed this article: *Chris Lee and Adith Swaminathan*

LIVE! 360
TECH EVENTS WITH PERSPECTIVE

SQL Server LIVE!

TRAINING FOR DBAs AND IT PROS

ROYAL PACIFIC RESORT AT UNIVERSAL ORLANDO
November 12-17

Data. Driven.

After 5 days of workshops, deep dives and breakout sessions, SQL Server Live! will leave you with the skills needed to drive data forward.

With timely, relevant content, SQL Server Live! helps administrators, DBAs, and developers do more with their SQL Server investment. Sessions will cover performance tuning, security, reporting, data integration, adopting new techniques, improving old approaches, and modernizing the SQL Server infrastructure.

REGISTER NOW

REGISTER BY SEPT 15 AND SAVE $400!*

Use promo code ORLSEP1

*Savings based on 5-day packages only. See website for details.

A Part of Live! 360: The Ultimate Education Destination

5 GREAT CONFERENCES, 1 GREAT PRICE

Visual Studio LIVE! | SQL Server LIVE! | TECHMENTOR | Office & SharePoint LIVE! | Modern Apps LIVE!

SQLLIVE360.COM

EVENT PARTNERS
Microsoft Magenic

PLATINUM SPONSOR
okta SMARTBEAR

SUPPORTED BY
Visual Studio msdn magazine Redmond Channel Partner
Redmond Magazine VIRTUALIZATION & Cloud Review Visual Studio Magazine

PRODUCED BY
1105 MEDIA YOUR GROWTH, OUR BUSINESS.

TED NEWARD

How To Be MEAN: Servicing Angular

Welcome back again, MEANers.

One of the common needs of an Angular application is to obtain or update data from locations not inside the browser—which is usually everywhere. Most often, an Angular app will need to exchange data (usually in the form of JSON packets) with an HTTP-based API implementation somewhere on the Internet.

Unfortunately, this doesn't really "fit" well with the notion of a component. Multiple components will often need to communicate with the same server through the same API implementation, and to duplicate that code in each and every component would get ugly. Certainly, as you've seen with the Upvote component (msdn.com/magazine/mt784667), it's possible to create a standalone TypeScript class and simply make use of that—essentially bringing the concept of "library module" into Angular. But as so frequently happens, the Angular team anticipated this need, and has given unto us … services.

A service, in Angular terms, is an injectable module: "module" because you'll create a TypeScript module that encapsulates the service's implementation, and "injectable" because through the use of some decorators, you'll make the service known to Angular, so that those components that wish to use it simply have to declare it as a parameter (similar to how the @Input/@Output decorators worked earlier) and lo, they will receive one.

(Take note: An Angular service is not implicitly in a 1-to-1 relationship with an "API service" or "microservice" or any of the other overloaded uses of the term. If the term is still confusing, remember that an Angular service is simply a TypeScript module whose users are always other Angular components—humans never interact with an Angular service directly.)

Let's have a look.

SpeakerService

The start of every service will typically begin with asking the Angular CLI to create the basic scaffolding for you, so let's start with that: From a command prompt, type "ng generate service Speaker." This will generate two files—speaker.service.ts and speaker.service.spec.ts—and inside of them will be the basic outline for the SpeakerService class (the "Service" is an assumed-desired suffix to the name specified on the command-line) and the testing code, respectively.

However, as of this moment, you don't have a Speaker type to be returned from the service! (As legendary cartoon character Homer Simpson would say … "D'oh!") Again, this is easily fixed by asking the CLI to create a Speaker type to use: "ng generate class Speaker." This time, the CLI will generate a single file, speaker.ts, which contains only the class Speaker. This is important to notice: The CLI is looking to enforce Angular conventions by automatically putting the appropriate suffix on the various types being generated. There's a command-line parameter to control the naming, but, frankly, just go with the defaults here—it will make many things much easier over time.

Speaker

I'm going to keep the Speaker type pretty straightforward for now. Obviously, this could get as large and as complicated as necessary, but it wouldn't teach you anything about Angular to do it that way. The Speakers have some simple properties, and you're going to take the simple (perhaps naïve) perspective that an evaluation is of the speaker, and not of the talk. Were this a production application, you'd probably have a "Speakers HAVE-MANY Talks" relationship and then "Talks HAVE-MANY Upvotes" relationship, but, again, that doesn't really show off Angular. So Speaker looks like this:

```
import { Upvote } from './upvote';

export class Speaker {
  id: number;
  firstName: string;
  lastName: string;
  votes: Upvote;
}
```

> A service, in Angular terms, is an injectable module.

It's a simple class, enough so that I'm going to skip showing the tests for it. (Note that if you go looking for the speaker.spec.ts file, it doesn't exist—by default, the CLI doesn't generate a test file for a simple class like this. If you want a test file—and you really, really should want one—then you should pass "—spec true" on the command line when doing the ng generate command.)

SpeakerService

Now that you have a type to represent speakers, you can build out a service that will basically act as a repository (meaning, following the basic concept of the Repository pattern) for speaker instances. For the moment, this will live in memory, and its (so far, read-only) implementation should be fairly clear, as you can see in **Figure 1**.

NEVRON

www.nevron.com

EXTENSIBLE INTERACTIVITY AND EDITING OPTIONS • AUTOMATIC GRAPH AND TREE LAYOUTS
IMPORT FROM DATABASE, VISIO AND ESRI DATA • HUNDREDS OF GENUINE CLIPART SHAPES
450+ EXAMPLES WITH SOURCE CODE

NEVRON DIAGRAM FOR .NET

ADVANCED & FAST

Part of NEVRON VISION for .NET

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

solution available for

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

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

Visual Studio Partner

The SpeakerService looks like pretty much any other TypeScript class, with the sole exception of the @Injectable decorator on it—this is what will tell Angular that the SpeakerService can be dependency injected into those components that want to use it. The rest of the implementation is pretty straightforward. Right now, everything is working off of a constant array of Speaker instances held as a "private global" inside the speaker.service.ts file, called "SPEAKERS," shown in **Figure 2**.

In effect, this array of three elements is your database, at least for the moment. When I talk about how to use Angular to send and receive HTTP requests, this array will be populated from JSON returned to use from the server's API implementation, and any changes you make will be to the array first. But for now, just hold it as a constant array.

Notice that the Upvote component is used directly inside the Speaker instances—this will allow the UpvoteComponent to build around the Upvote instances, and essentially preserve the whole MVC approach even at a fractal level inside models. Spiffy.

Figure 1 **Implementing the SpeakerService**

```
import { Injectable } from '@angular/core';

import { Speaker } from './speaker';
import { Upvote } from './upvote';

@Injectable()
export class SpeakerService {
  constructor() { }
  getSpeakers() : Array<Speaker> {
    return SPEAKERS;
  }
  getSpeaker(id: number) : Speaker {
    return SPEAKERS.find(speaker => speaker.id === id);
  }
}
```

Figure 2 **The Constant Array of Speaker Instances**

```
const SPEAKERS : Array<Speaker> = [
  {
    id: 0,
    firstName: "Ted",
    lastName: "Neward",
    votes: new Upvote(30)
  },
  {
    id: 1,
    firstName: "Rachel",
    lastName: "Appel",
    votes: new Upvote(35)
  },
  {
    id: 2,
    firstName: "Nick",
    lastName: "Landry",
    votes: new Upvote(3)
  },
];
```

Figure 3 **Using the SpeakerService**

```
import { Component } from '@angular/core';

import { Speaker } from './speaker';
import { SpeakerService } from './speaker.service';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'SpeakerApp';
  speaker: Speaker;

  constructor(private speakerSvc: SpeakerService)
  {
    this.speaker = this.speakerSvc.getSpeaker(0);
  }
}
```

Using SpeakerService

The UI currently isn't much to speak of (pun intended), but let's look at how to make use of the service anyway. Assume that the homepage simply wants to grab the first Speaker out of the SpeakerService (using the getSpeaker call with a parameter of "0") and then extract the Upvote out of it for use in the UpvoteComponent on that page. Practically speaking, that means that the app.component file will need to get a reference to a SpeakerService, use that to obtain a Speaker, and then get a reference to the Upvote out of the Speaker and pass that to the UpvoteComponent. The latter part is pretty easy—you just modify the app.component.html template to reference an AppComponent field called speaker, from which you'll get the Upvote instance, like so:

```
<h1>
  {{title}}
</h1>

<app-upvote votes="{{speaker.votes}}"><b>Current Upvotes:</b></app-upvote>
```

This is because Angular needs one more bit of glue in order to do the @Injectable magic. It needs to know that the code in this file is actually something that it's supposed to know about.

So far, so good. But this means that you need a local speaker field inside the AppComponent, and that's where you'll also use the SpeakerService, as shown in **Figure 3**.

Wow, this Angular stuff is actually pretty easy. Save everything, do an "ng serve" and … blank screen. What gives?

Providing SpeakerServices

Opening up the developer tools console in your browser yields a hint: At the top of the stack trace Angular says "No provider for SpeakerService!" in a rather loud voice. This is because Angular needs one more bit of glue in order to do the @Injectable magic. It needs to know that the code in this file is actually something that it's supposed to know about. By default, Angular doesn't just assume every .ts file in the directory is part of the application, so you need to set up a "provider" for the SpeakerService. This is done in the

Spreadsheets Made Easy.

SpreadsheetGear 2017 Released

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

Scalable Reporting

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

.NET Standard

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

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

WIN Windows Forms | Silverlight | WPF WPF

Powerful Controls

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

Comprehensive Charting

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

Fastest Calculations

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

Download your free fully functional evaluation at SpreadsheetGear.com

SpreadsheetGear

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

Figure 4 **Set Up SpeakerService in the Providers Array**

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { HttpModule } from '@angular/http';

import { AppComponent } from './app.component';
import { UpvoteComponent } from './upvote/upvote.component';
import { SpeakerService } from './speaker.service';

@NgModule({
  declarations: [
    AppComponent,
    UpvoteComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
    HttpModule
  ],
  providers: [ SpeakerService ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

AppModule (app.module.ts), and it doesn't require a rocket scientist to figure out where. There's an empty "providers" array right in the AppModule's NgModule decorator, just begging to have SpeakerService listed there (after doing the appropriate import to reference the speaker.service.ts file, of course). **Figure 4** has the goods.

Save, run (or just wait for the browser to refresh if you still have "ng serve" running in the command-line window) and look at the loveliness.

Property Bindings

Except … it's not lovely. Matter of fact, if you're running the code, you're probably noticing that instead of a number (30), the page is displaying something like Current Upvotes: [object Object]. What's up with that?

> Opening up the developer tools console in your browser yields a hint: At the top of the stack trace Angular says "No provider for SpeakerService!" in a rather loud voice.

If you look back at the UpvoteComponent template, you'll notice that it's using the {{ }} syntax to obtain a value for the HTML:

```
<app-upvote model="{{speaker.votes}}"><b>Current Upvotes:</b></app-upvote>
```

You could get away with it for a while, but in this particular case, because you're now handing in an actual object (instead of the earlier numeric constant), that expression is evaluating to an object, and then converted into a string before being passed in as the input to the UpvoteComponent. Bad column author! No pizza!

What the code should've been doing all along is using a property-binding declaration instead of the string interpolation syntax. Fortunately, this is easily fixed:

```
<app-upvote [model]="speaker.votes"><b>Current Upvotes:</b></app-upvote>
```

> The expression will be evaluated, and the result bound to the property in question.

Note the square brackets around the name of the property you want to bind to, and notice how the value of the property-bind is a TypeScript expression to use. The expression will be evaluated, and the result bound to the property in question—in this case, the "model" property of the UpvoteComponent. (I chose to rename the "votes" property to "model" because that's an emerging convention I'm using for my Angular code: For a given component, its model should be stored in a property/field called model, so I can keep straight what the principal state is, as opposed to incidental properties that might be used during the component's lifetime.) Once the binding takes place, instead of interpolating the object to the ECMAScript default [object Object] and then passing that in, the object reference itself will be sent to the UpvoteComponent, which recognizes it as an Upvote, and everything is back to loveliness again.

Lesson learned!

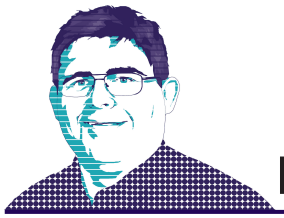
Wrapping Up

Things are starting to fall into place now—components are slowly taking shape around an MVC concept, and the models are being retrieved by services. You still need to get those services to obtain real data (as opposed to faking it, the way it's being done now), and you still need some kind of ubiquitous master-detail "drill-down" display where you begin with a list of speakers and select one for more details, not to mention the ability to edit speaker details and add new speakers to the list. But, things are starting to hang together better now.

After the work done here, you can start to see how the component-based approach works collectively to "build up" an application out of mostly self-contained components. The more componentization that can be put into place, the easier it will be to test those components, and the more testing that can be done, the more the application will resist stupid programmer mistakes during its overall development.

There's still more work to do … for now … happy coding! ■

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



Protocol Registration and Activation in UWP Apps

In my last column, I explored how to use the Launcher class to extend the functionality of a Universal Windows Platform (UWP) app by leveraging the features of other apps built into Windows. For example, I was able to add fully featured mapping functionality, such as searching for a good place to find pizza and finding directions complete with traffic reports, without having to add significant complexity to my app's code base. In this month's column, I'll explore the other side of the equation: How to expose a UWP app's functionality to other apps.

Of course, programmatically launching applications and passing parameters along from other applications predates the UWP. In fact, many Windows apps support this via command-line switches. For example, to launch Microsoft Word in Safe Mode, where add-ons and plug-ins are disabled, pass the parameter "/safe" to the WinWord.exe file. To test this out, press Windows Key+R, to bring up the Run dialog box. In the textbox, type "Winword.exe /safe" and click OK. If you have Microsoft Word installed on your system, then the program will launch in Safe Mode. Additionally, similar commands exist for Excel, PowerPoint and others. Clearly, launching a program with parameters isn't a new mechanism. What's new, however, is the additional mechanism that UWP provides: Uniform Resource Identifier (URI) activation.

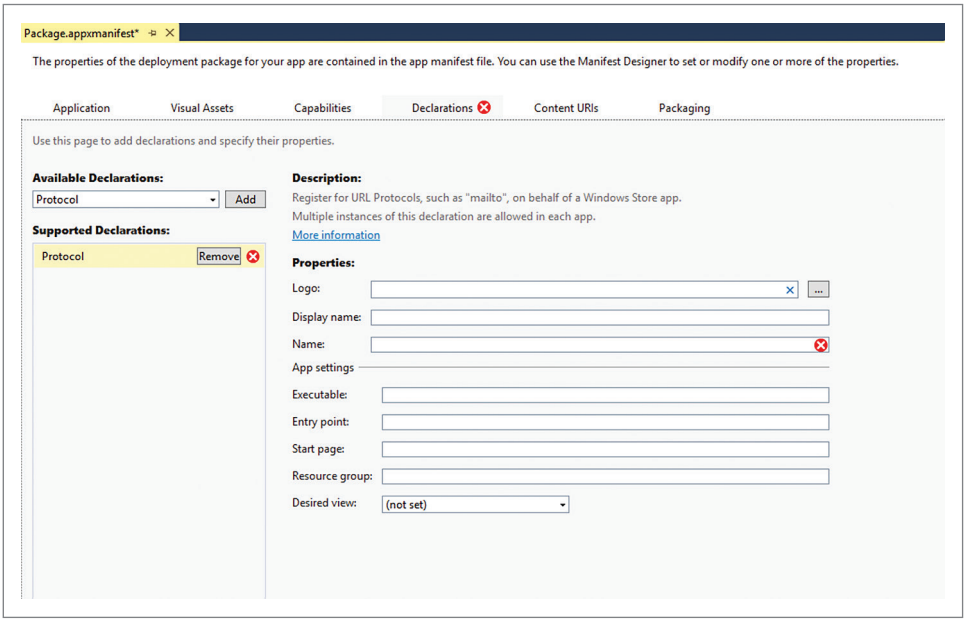


Figure 1 **Adding a Protocol Declaration (Note the Validation Errors)**

URI Activation

In my last column (msdn.com/magazine/mt784669), I used URI activation to launch and pass parameters to several apps. The apps then took these input parameters and acted upon them. For example, to get driving directions between Washington, D.C., and New York City with the default Maps app in Windows 10, the app simply passes the following URI to the Launcher class:

```
bingmaps:?rtp=adr.Washington,%20DC~adr.New%20York,%20
NY&amp;mode=d&amp;trfc=1
```

Code download available at bit.ly/2gOkEEZ.

Alternatively, typing this command into the Run dialog box will also yield the same result. The Maps app has been registered to handle the "bingmaps" protocol. The text after the question mark represents parameters passed along to the Maps app. In this case, it sets the start and end points, along with a mode to indicate driving ("d") and a 1 to add traffic data to the results.

What if you wanted your app to perform in a similar manner? What are the steps you'd need to take in order to register a protocol and handle the subsequent parameters passed to it?

The first step is to create a new project. Create a new blank UWP project in Visual Studio by choosing New Project from the File menu. Expand Installed Templates | Windows | Blank App (Universal Windows). Name the project ProtocolActivation and then click OK. Immediately afterward, a dialog box will appear asking you which version of Windows the app should target. For this project, the default options will be fine, so you can simply click OK.

Registering a URI Protocol

Now that the project has been created, open the appmanifest to register a protocol. In the Solution explorer, look for the Package.appmanifest file and double-click on it to open it. Click on the Declarations tab. Choose Protocol from the dropdown underneath Available

Declarations. Now, click Add to add the declaration to the manifest. The screen should look similar to **Figure 1**.

While it's highly recommended to provide entries for the Logo and Display name fields, the only required field is Name. Enter "bingmaps" into the textbox. Press the Tab key. Note that the validation errors disappear after the textbox loses focus. Run the project by pressing F5, choosing Start Debugging from the Debug Menu, or simply clicking on the play icon on the toolbar. Once the app is running, press Windows Key+R to launch the Run dialog. Enter "bingmaps:" into the textbox and click OK. Note that the default Maps app started just as before.

While highly recommended to provide entries for the Logo and Display name fields, the only required field is Name.

This brings up an important caveat: Some protocols cannot be overridden. For a complete list, either refer to my previous column or refer to the MSDN documentation on Reserved File and URI Scheme Names at bit.ly/2st28Er. Choosing any of these reserved names will not result in an error message; the app simply won't get activated.

Registering a Custom Protocol

Close the app if it's still running. Go back into the manifest file and change the contents of the Name field to "msdncolors" and save the project. Now, press Windows Key+R to bring up the Run dialog, enter "msdncolors" into the textbox and click OK. A system-wide dialog box appears suggesting to look for an app in the Store that can handle this protocol, as shown in **Figure 2**.

While the protocol might have been declared in the app's manifest file, it hasn't yet been registered on the system. This happens at install time. Run the solution now and then enter "msdncolors" into the Run dialog once again. The same dialog box appears, but with a new option added: the ProtocolActivation app, as shown in **Figure 3**. Click OK. The ProtocolActivation app will launch. Close the app. Enter "msdncolors" once more into the Run dialog box. This time, there will be no dialog box, the app will simply run whether the "Always use this app" option was checked.

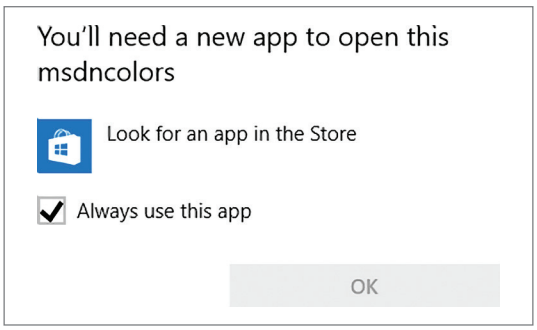


Figure 2 **Dialog Box That Appears When the System Sees an Unfamiliar Protocol**

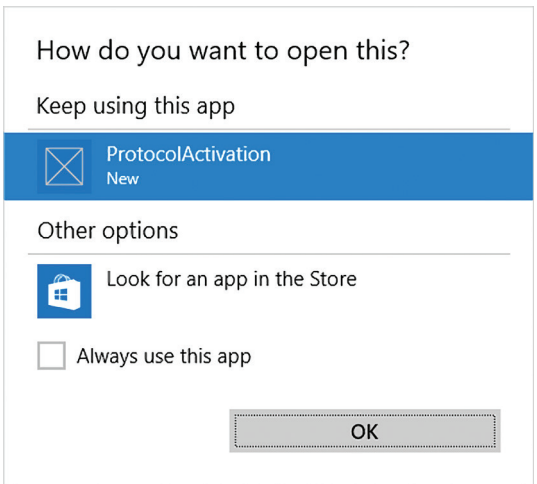


Figure 3 **The New App Now Appears as an Option**

Handling Activation

UWP apps can be activated any number of ways, from being launched by the user when its tile is tapped or when activated by protocol. In order to detect how the app was launched, the OnActivated event handler must be overridden.

In Solution Explorer, open the App.xaml.cs file, and add the following code to the App class:

```
protected async override void OnActivated(IActivatedEventArgs e)
{
  if (e.Kind == ActivationKind.Protocol)
  {
    var dialog = new MessageDialog("App Activated by Protocol.");
    await dialog.ShowAsync();
  }
  Window.Current.Activate();
}
```

Run the solution once again, open the Run dialog, type in "msdncolors:" and click OK. This time, you'll see a message dialog. Stop the app from running. Now, re-open the Run dialog by pressing Windows Key+R, enter "msdncolors:" into the textbox and click OK. Note that the same message dialog appears, but the app doesn't progress beyond the splash screen. The reason why will be addressed later.

Passing Parameters via URI

While launching an app through URI activation has its uses, the real power comes in passing parameters from one app to another. This can be done by appending parameters to the protocol request. This mechanism operates nearly the same way as HTTP GET requests.

For example, URIs follow a pattern of: [protocol]:[host address]?[parameter1]=[value1]&[parameter2]=[value2]

For instance, given the URI http://bing.com/search?q=data%20driven%20podcast, the protocol is HTTP, the host address is bing.com, and the value of "data%20driven%20podcast" is passed to the parameter "q." Note that the spaces in "data driven podcast" are converted to "%20." Parameter names and values passed along in a URI must be encoded.

Modify the code inside the OnActivated method to contain the changes shown in **Figure 4**.

The code in **Figure 4** casts the IActivatedEventArgs parameter to a ProtocolActivatedEventArgs class, providing access to a number of properties specific to protocol activation scenarios. In this instance, I'm mainly concerned with the URI property. The URI will contain the information passed to the app via protocol activation. Run the app again, open the Run dialog, enter "msdncolors:background?red" into the textbox, and

click OK. Once again, the app will launch. This time, however, the Message dialog will have the Query String and AbsolutePath values displayed.

Stop the app and re-run the same command from the Run dialog. Once again, the app doesn't progress beyond the splash screen. It's time to address that.

Figure 4 **Modifying the Code Inside the OnActivated Method**

```
if (e.Kind == ActivationKind.Protocol)
{
  var args = e as ProtocolActivatedEventArgs;
  var message = "no parameters passed";
  if (args.Uri.PathAndQuery != string.Empty)
  {
    var queryString = args.Uri.Query;
    var absolutePath = args.Uri.AbsolutePath;
    message = $"Query String: {queryString}\n AbsolutePath: {absolutePath}";
  }
  var dialog = new MessageDialog(message);
  await dialog.ShowAsync();
}
Window.Current.Activate();
```

Figure 5 **Updated OnActivated Code**

```
protected override void OnActivated(IActivatedEventArgs e)
{
  if (e.Kind == ActivationKind.Protocol)
  {
    var args = e as ProtocolActivatedEventArgs;
    if (args.Uri.PathAndQuery != string.Empty)
    {
      Frame rootFrame = Window.Current.Content as Frame;
      if (rootFrame == null)
      {
        rootFrame = new Frame();
        rootFrame.NavigationFailed += OnNavigationFailed;
        Window.Current.Content = rootFrame;
      }
      if (rootFrame.Content == null)
      {
        if (!rootFrame.Navigate(typeof(MainPage)))
        {
          throw new Exception("Failed to create initial page");
        }
      }
      var fragment = args.Uri.Fragment;
      var absolutePath = args.Uri.AbsolutePath;
      var mainPage = rootFrame.Content as MainPage;
      mainPage.NavigateWithParameters(absolutePath, fragment);
    }
  }
  Window.Current.Activate();
}
```

Figure 6 **The NavigateWithParameters Method**

```
public void NavigateWithParameters(string target, string color)
{
  string hexCode = color.Replace("#", string.Empty);
  byte a = (byte)(Convert.ToUInt32(hexCode.Substring(0, 2), 16));
  byte r = (byte)(Convert.ToUInt32(hexCode.Substring(2, 2), 16));
  byte g = (byte)(Convert.ToUInt32(hexCode.Substring(4, 2), 16));
  byte b = (byte)(Convert.ToUInt32(hexCode.Substring(6, 2), 16));

  var brush = new SolidColorBrush(Color.FromArgb(a, r, g, b));

  if (target.ToLower() == "background")
  {
    grdBackground.Background = brush;
  }
  else
  {
    grdForeground.Background = brush;
  }
}
```

Navigating to a Page

Modify the OnActivated method in App.xaml.cs to match the code in **Figure 5**. While this code looks complex, it's actually not. Much of the additional code concerns revolve around finding the current content of the active window. In cases where the app is already running, the current content for this app will be the MainPage class. When the app isn't running and is launched by protocol activation, there's no current content, and Window.Current.Content has a null value. In those cases, the code creates a new Frame, sets the appropriate event handlers and content, and navigates to the MainPage. For more information about the Window and Frame classes, refer to bit.ly/2tGwt1H and bit.ly/2sQ8LTY, respectively.

The rest of the code is a slight modification from before, with the addition of making a call to a NavigateWithParameters method. This is how the information from the URI will get sent to the MainPage class. Now is the time to add that method to the MainPage.xaml.cs file and add some UI elements to the MainPage.xaml file.

First, modify the MainPage.xaml file to have the following XAML inside the Page element:

```
<Grid Name="grdBackground" Background=
  "{ThemeResource ApplicationPageBackgroundThemeBrush}">
  <Grid Name="grdForeground" Width="100" Height="100"></Grid>
</Grid>
```

This adds the name "grdBackground" to the Grid control already present and adds an inner grid named "grdForegound."

Next, add the code in **Figure 6** to the MainPage.xaml.cs file. The bulk of the code converts a hex code ARGB color value like #FFAA3CC7 into something that a SolidColorBrush can use. The conversion code drew inspiration from the blog post at bit.ly/2tGuFFH, which also contains a great explanation of how the conversion works. The remainder of the code is fairly simple; it changes the color of one grid or the other based on the target parameter.

Once all the code is in place, run the app. In the Run dialog box type "msdncolors:foreground?#FFAA3CC7" and click OK. This should turn the foreground grid a shade of purple. Now, type "msdncolors:background?#FFCFCFCF" into the textbox of the Run dialog and click OK. The background Grid should turn a light gray. Close the app. Now, re-enter the previous commands to see that the app is the same when it's not already running.

Wrapping Up

In this column, I demonstrated how to register a UWP app to handle a particular protocol. Adding this feature will make your UWP apps accessible to other apps on a user's device. This lets other developers create interesting uses for the apps you develop and makes your app more desirable. While much of the activation in this column was done using the Run dialog, passing the same URI to the Launcher class will yield the same results. ■

Frank La Vigne *is chief evangelist at DataLeader.IO, where he helps customers leverage technology in order to create a better world. He is co-host of the data science podcast Data Driven (DataDriven.tv) and blogs regularly at FranksWorld.com.*

Thanks to the following technical expert for reviewing this article: *Jose Luis Manners*

Visual Studio LIVE!

EXPERT SOLUTIONS FOR .NET DEVELOPERS

CHICAGO
SEPTEMBER 18 - 21, 2017
DOWNTOWN MARRIOTT

INTENSE TRAINING FOR DEVELOPERS, ENGINEERS, PROGRAMMERS, ARCHITECTS AND MORE!

Development Topics Include:

- Visual Studio / .NET Framework
- JavaScript / HTML5 Client
- Native Client
- Software Practices
- Database and Analytics
- Angular JS
- ASP.NET / Web Server
- Agile
- ALM / DevOps
- Cloud Computing
- Windows Client
- Xamarin

Register By Sept 18 and Save $200!

MUST USE DISCOUNT CODE CHEB01

REGISTER NOW

GOLD SPONSOR

Suncfusion

SUPPORTED BY

Visual Studio msdn magazine Visual Studio MAGAZINE

PRODUCED BY

1105 MEDIA

CHICAGO AGENDA AT-A-GLANCE

ALM / DevOps	Cloud Computing	Database and Analytics	Native Client	Software Practices	Visual Studio / .NET Framework	Web Client	Web Server

START TIME	END TIME				
		Visual Studio Live! Pre-Conference Workshops: Monday, September 18, 2017 *(Separate entry fee required)*			
7:30 AM	9:00 AM	Pre-Conference Workshop Registration - Coffee and Morning Pastries			
9:00 AM	6:00 PM	**M01** Workshop: Distributed Cross-Platform Application Architecture - *Jason Bock & Rockford Lhotka*	**M02** Workshop: Practical ASP.NET DevOps with VSTS or TFS - *Brian Randell*	**M03** Workshop: SQL Server 2016 for Developers - *Andrew Brust & Leonard Lobel*	
6:45 PM	9:00 PM	Dine-A-Round			
START TIME	**END TIME**	**Visual Studio Live! Day 1: Tuesday, September 19, 2017**			
7:00 AM	8:00 AM	Registration - Coffee and Morning Pastries			
8:00 AM	9:00 AM	KEYNOTE: Modern Application Development with Visual Studio, .NET and Azure - Jay Schmelzer, *Director of Program Management, Visual Studio Team, Microsoft*			
9:15 AM	10:30 AM	**T01** JavaScript for the C# Developer - *Philip Japikse*	**T02** Storyboarding 101 - *Billy Hollis*	**T03** Build Cross-Platform Apps in C# using CSLA .NET - *Rockford Lhotka*	**T04** What's New in Visual Studio 2017 - *Robert Green*
10:45 AM	12:00 PM	**T05** TypeScript: The Future of Front End Web Development - *Ben Hoelting*	**T06** Better, Faster, Automated! Windows App Deployment in Visual Studio Mobile Center - *Ela Malani & Piyush Joshi*	**T07** Roll Your Own Dashboard in XAML - *Billy Hollis*	**T08** What's New in C#7 - *Jason Bock*
12:00 PM	1:00 PM	Lunch - Visit Exhibitors			
1:00 PM	1:30 PM	Dessert Break - Visit Exhibitors			
1:30 PM	2:45 PM	**T09** ASP.NET Core MVC - What You Need to Know - *Philip Japikse*	**T10** Professional Scrum Development Using Visual Studio 2017 - *Richard Hundhausen*	**T11** What's New for Developers in SQL Server 2016 - *Leonard Lobel*	**T12** To Be Announced
3:00 PM	4:15 PM	**T13** User Authentication for ASP.NET Core MVC Applications - *Brock Allen*	**T14** PowerShell for Developers - *Brian Randel*	**T15** What's New in Azure IaaS v2 - *Eric D. Boyd*	**T16** To Be Announced
4:15 PM	5:30 PM	Welcome Reception			
START TIME	**END TIME**	**Visual Studio Live! Day 2: Wednesday, September 20, 2017**			
7:30 AM	8:00 AM	Registration - Coffee and Morning Pastries			
8:00 AM	9:15 AM	**W01** Securing Web APIs in ASP.NET Core - *Brock Allen*	**W02** Professional Software Testing Using Visual Studio 2017 - *Richard Hundhausen*	**W03** Cloud Oriented Programming - *Vishwas Lele*	**W04** Database Development with SQL Server Data Tools - *Leonard Lobel*
9:30 AM	10:45 AM	**W05** Assembling the Web - A Tour of WebAssembly - *Jason Bock*	**W06** Get Started with Git and GitHub - *Robert Green*	**W07** Building Modern Web Apps with Azure - *Eric D. Boyd*	**W08** Tactical DevOps for SQL Server - *Brian Randell*
11:00 AM	12:00 PM	General Session: To Be Announced			
12:00 PM	1:00 PM	Birds-of-a-Feather Lunch - Visit Exhibitors			
1:00 PM	1:30 PM	Dessert Break - Visit Exhibitors - Exhibitor Raffle @ 1:15pm (Must be present to win)			
1:30 PM	2:45 PM	**W09** Tools for Modern Web Development Dev Ops - *Ben Hoelting*	**W10** Go Mobile with C#, Visual Studio, and Xamarin - *James Montemagno*	**W11** Build Awesome AF Apps! - *Rachel Appel*	**W12** Power BI: Analytics for Desktop, Mobile and Cloud - *Andrew Brust*
3:00 PM	4:15 PM	**W13** Get Rid of HTML Tables for Better Mobile Web Applications - *Paul Sheriff*	**W14** Continuous Integration & Deployment for Mobile Apps - *James Montemagno*	**W15** Microservices with Azure Container Service & Service Fabric - *Vishwas Lele*	**W16** Power BI: Beyond the Basics - *Andrew Brust*
4:30 PM	5:45 PM	**W17** Use HTML5/Bootstrap to Build Business UI - *Paul Sheriff*	**W18** Mobilizing your Existing Enterprise Applications - *Nick Landry*	**W19** Busy Developer's Guide to the Google Cloud Platform - *Ted Neward*	**W20** Big Data Solutions in Azure - *David Giard*
7:00 PM	9:00 PM	Visual Studio Live! Evening Event			
START TIME	**END TIME**	**Visual Studio Live! Day 3: Thursday, September 21, 2017**			
7:30 AM	8:00 AM	Web Client			
8:00 AM	9:15 AM	**TH01** I Just Met You, and "This" is Crazy, But Here's My NaN, So Call(Me), Maybe? - *Rachel Appel*	**TH02** PowerApps, Flow, and Common Data Service: Empowering Businesses with the Microsoft Business Application Platform - *Archana Nair*	**TH03** Lessons Learned from Real World Xamarin.Forms Projects - *Nick Landry*	**TH04** Busy Developer's Guide to NoSQL - *Ted Neward*
9:30 AM	10:45 AM	**TH05** Build Object-Oriented Enterprise Apps in JavaScript with TypeScript - *Rachel Appel*	**TH06** PowerApps and Flow Part II: Package, Embed, and Extend Your Applications - *Anousha Mesbah & Pratap Ladhani*	**TH07** Improve Your Retrospective Outcomes with Agile Kaizen - *Angela Dugan*	**TH08** Building Applications with DocumentDb - New Features and Best Practices - *Raj Krishnan*
11:00 AM	12:15 PM	**TH09** What's New in TypeScript? - *Doris Chen*	**TH10** Getting Started with Entity Framework Core - *Jim Wooley*	**TH11** Open Source for Microsoft Developers - *Rockford Lhotka*	**TH12** Introduction to Machine Learning with R - *Raj Krishnan*
12:15 PM	1:15 PM	Lunch			
1:15 PM	2:30 PM	**TH13** Practical Performance Tips and Tricks to Make Your HTML/ JavaScript Faster - *Doris Chen*	**TH14** Getting Your Agile Team Unstuck! Tips and Tricks for Blasting Through Common Setbacks - *Angela Dugan*	**TH15** DI Why? Getting a Grip on Dependency Injection - *Jeremy Clark*	**TH16** The Rise of the Machines - Machine Learning for Developers - *Adam Tuliper*
2:45 PM	4:00 PM	**TH17** Building Powerful Applications with AngularJS 2 and TypeScript - *David Giard*	**TH18** Improving Code Quality with Roslyn Code Analyzers - *Jim Wooley*	**TH19** Unit Testing Makes Me Faster: Convincing Your Boss, Your Co-Workers, and Yourself - *Jeremy Clark*	**TH20** I'm Emotional - Using Microsoft Cognitive Services to Understand the World Around You - *Adam Tuliper*

Speakers and sessions subject to change

vslive.com/chicagomsdn

CONNECT WITH US

twitter.com/vslive – @VSLive

facebook.com – Search "VSLive"

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

Visual Studio LIVE!

EXPERT SOLUTIONS FOR .NET DEVELOPERS

ANAHEIM, CA
OCT 16-19, 2017
HYATT REGENCY
A Disneyland® Good Neighbor Hotel

INTENSE TRAINING FOR DEVELOPERS, ENGINEERS, PROGRAMMERS, ARCHITECTS AND MORE!

Development Topics include:

- Visual Studio / .NET
- JavaScript / HTML5
- Angular
- Native Mobile & Xamarin
- Software Practices
- Database and Analytics
- ASP.NET Core
- Web API
- ALM / DevOps
- Cloud Computing
- UWP
- Unit Testing

Register by Sept 22 and Save $200!

Use promo code VSLAN2

REGISTER NOW

EVENT PARTNER
Microsoft

SUPPORTED BY
Visual Studio msdn magazine Visual Studio MAGAZINE

PRODUCED BY
1105 MEDIA
YOUR GROWTH. OUR BUSINESS.

ANAHEIM AGENDA AT-A-GLANCE

ALM / DevOps | Cloud Computing | Database and Analytics | Native Client | Software Practices | Visual Studio / .NET Framework | Web Client | Web Server

START TIME	END TIME				
Visual Studio Live! Pre-Conference Workshops: Monday, October 16, 2017 *(Separate entry fee required)*					
7:30 AM	9:00 AM	Pre-Conference Workshop Registration - Coffee and Morning Pastries			
9:00 AM	6:00 PM	**M01** Workshop: Distributed Cross-Platform Application Architecture - *Jason Bock & Rockford Lhotka*	**M02** Workshop: Practical ASP.NET DevOps with VSTS or TFS - *Brian Randell*	**M03** Workshop: Developer Dive into SQL Server 2016 - *Leonard Lobel*	
6:45 PM	9:00 PM	Dine-A-Round			
START TIME	**END TIME**	**Visual Studio Live! Day 1: Tuesday, October 17, 2017**			
7:00 AM	8:00 AM	Registration - Coffee and Morning Pastries			
8:00 AM	9:00 AM	KEYNOTE: To Be Announced			
9:15 AM	10:30 AM	**T01** What's New in TypeScript? - *Doris Chen*	**T02** Go Mobile with C#, Visual Studio, and Xamarin - *James Montemagno*	**T03** To Be Announced	**T04** What's New in Visual Studio 2017 - *Robert Green*
10:45 AM	12:00 PM	**T05** Build Object-Oriented Enterprise Apps in JavaScript with TypeScript - *Rachel Appel*	**T06** Optimizing and Extending Xamarin.Forms Mobile Apps - *James Montemagno*	**T07** What's New for Developers in SQL Server 2016 - *Leonard Lobel*	**T08** To Be Announced
12:00 PM	1:30 PM	Lunch			
1:30 PM	2:45 PM	**T09** Angular 101: Part 1 - *Deborah Kurata*	**T10** Get Started with Git and GitHub - *Robert Green*	**T11** Exploring T-SQL Enhancements: Windowing and More - *Leonard Lobel*	**T12** Microsoft Teams - More Than Just Chat! - *Nedra Allmond*
3:00 PM	4:15 PM	**T13** Angular 101: Part 2 - *Deborah Kurata*	**T14** Do It Again, Faster! Automate Your Windows Deployment Pipeline - *Ela Malani*	**T15** What's New in Azure IaaS v2 - *Eric D. Boyd*	**T16** Open Source for the Microsoft Developer - *Rockford Lhotka*
4:15 PM	5:30 PM	Welcome Reception			
START TIME	**END TIME**	**Visual Studio Live! Day 2: Wednesday, October 18, 2017**			
7:00 AM	8:00 AM	Registration - Coffee and Morning Pastries			
8:00 AM	9:15 AM	**W01** I Just Met You, and "This" is Crazy, But Here's My NaN, So Call(Me), Maybe? - *Rachel Appel*	**W02** Tactical DevOps with VSTS - *Brian Randell*	**W03** Go Serverless with Azure Functions - *Eric D. Boyd*	**W04** What's New in C#7 - *Jason Bock*
9:30 AM	10:45 AM	**W05** Practical Performance Tips and Tricks to Make Your HTML/JavaScript Faster - *Doris Chen*	**W06** Real World VSTS Usage for the Enterprise - *Jim Szubryt*	**W07** Cloud Oriented Programming - *Vishwas Lele*	**W08** I'll Get Back to You: Understanding Task, Await, and Asynchronous Methods - *Jeremy Clark*
11:00 AM	12:00 PM	General Session: The Future of Technology Seen through the Eyes of Computer Vision - *Founder / Exec Chairman, InterKnowlogy, Actus, VSBLTY*			
12:00 PM	1:30 PM	Birds-of-a-Feather Lunch			
1:30 PM	2:45 PM	**W09** Assembling the Web - A Tour of WebAssembly - *Jason Bock*	**W10** Database Lifecycle Management and the SQL Server Database - *Brian Randell*	**W11** Microservices with Azure Container Service & Service Fabric - *Vishwas Lele*	**W12** Getting Started with Entity Framework Core - *Jim Wooley*
3:00 PM	4:15 PM	**W13** Building Single Page Web Applications Using Aurelia.js and the MVVM Pattern - *Ben Hoelting*	**W14** Getting to SAFe in the Enterprise - *Jim Szubryt*	**W15** Busy Developer's Guide to the Clouds - *Ted Neward*	**W16** Improving Code Quality with Roslyn Code Analyzers - *Jim Wooley*
4:30 PM	05:45 PM	**W17** Securing Angular Apps - *Brian Noyes*	**W17** Building Applications with DocumentDb - New Features and Best Practices - *Raj Krishnan*	**W18** Busy Developer's Guide to the Google Cloud Platform - *Ted Neward*	**W19** Learn to Love Lambdas (and LINQ, Too) - *Jeremy Clark*
7:00 PM	9:00 PM	Visual Studio Live! Evening Event			
START TIME	**END TIME**	**Visual Studio Live! Day 3: Thursday, October 19, 2017**			
7:30 AM	8:00 AM	Registration - Coffee and Morning Pastries			
8:00 AM	9:15 AM	**TH01** ASP.NET Core MVC - What You Need to Know - *Philip Japikse*	**TH02** Tools for Modern Web Development Dev Ops - *Ben Hoelting*	**TH03** The Rise of the Machines - Machine Learning for developers - *Adam Tuliper*	**TH04** Storyboarding 101 - *Billy Hollis*
9:30 AM	10:45 AM	**TH05** Role-Based Security Stinks: How to Implement Better Authorization in ASP.NET and ASP.NET Core - *Benjamin Day*	**TH06** Building Cross-Platform Apps in C# using CSLA .NET - *Rockford Lhotka*	**TH07** Introduction to Machine Learning with R - *Raj Krishnan*	**TH08** Agile: You Keep Using That Word... - *Philip Japikse*
11:00 AM	12:15 PM	**TH09** From Zero to the Web API - *Paul Sheriff*	**TH10** Programming with the Model-View-ViewModel Pattern - *Miguel Castro*	**TH11** I'm Emotional - Using Microsoft Cognitive Services to Understand the World Around You - *Adam Tuliper*	**TH12** Top 10 Ways to Go from Good to Great Scrum Master - *Benjamin Day*
12:15 PM	1:15 PM	Lunch			
1:15 PM	2:30 PM	**TH13** Cortana Everywhere: Speech, Conversation & Skills Development - *Nick Landry*	**TH14** Roll Your Own Dashboard in XAML - *Billy Hollis*	**TH15** Unit Testing T-SQL Code - *Steve Jones*	**TH16** Exposing an Extensibility API for your Applications - *Miguel Castro*
2:45 PM	4:00 PM	**TH17** Securing Web Apps and APIs with IdentityServer - *Brian Noyes*	**TH18** Windows 10 for Developers: Building Universal Apps for 1B Devices - *Nick Landry*	**TH19** A Tour of SQL Server Security Features - *Steve Jones*	**TH20** Real World Applications for Dependency Injection - *Paul Sheriff*

Speakers and sessions subject to change

vslive.com/anaheimmsdn

CONNECT WITH US

twitter.com/vslive – @VSLive

facebook.com – Search "VSLive"

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

DAVID S. PLATT

ANNABELLE ROSE PLATT

When Software Sucks

I've had it with high school, and I'm only halfway through. Specifically, I've had it with high school software. Let me explain.

I'm Dave's daughter. I'm 17 years old and a digital native (see Dave's June 2015 column at msdn.com/magazine/mt147245). I've grown up with technology, so I don't expect to struggle with it. And just like my dad, I get angry when I see bad software, and furious when I'm expected to use it. You should, too.

The people who write software used in high schools don't seem to care much about making it good. And the school doesn't realize that ill-designed software can be a dangerous tool that works against the very things it hopes to support. That's probably because the administration is comprised of [nominal] adults. As digital immigrants, they've always struggled with software, and assume that's natural and necessary. My 10th grade algebra II class illustrates this principle quite well.

If you're not familiar with Pearson, you don't work in education and you should congratulate yourself on both of those accomplishments. You'll probably live a few years longer than the rest of us. Pearson is a publishing company that's now churning out one bad piece of educational software after another.

I'm specifically referring to pearsonrealize.com, which publishes our online math textbook platform. The teacher would assign us problems to complete on the platform itself, where she could grade and return them. It sounds like a great idea, if executed well. Which it's not.

My class hated these online assignments. I spent more time typing in my answers than doing the actual problems. When I got a problem wrong, it was usually because of formatting, not my calculations. Every character must be formatted specifically. For just one example, if you're typing in an ordered pair, the answer must be written as an open parenthesis, number, comma, space, number, close parenthesis. If you forget that space, buddy, your answer is wrong.

This wild goose chase takes a while to figure out. Teachers rarely know how to use the software (that digital native/immigrant thing again), so it's left to the one person that's nosed around a bit and figured out whatever convoluted method the programmers have decided to call "intuitive" to instruct the rest of the class. Usually this is me. It's aggravating to tell 25 different people the same process over and over again. Dave says I should charge for tech support.

To make matters worse, we used iPads in class, and Pearson hasn't optimized its interface for touch-dependent devices. In 2017? Seriously? When you tap the ordered-pair button, the correctly formatted ordered pair comes up, with highlighted blue spaces that look like you should just type your answer in. But that would be too easy. Instead, you have to delete those spaces (not the commas though!), then type your answer in instead. In a tiny box, on an iPad, with your clumsy finger. I wanted to gouge my eyes out with a spoon.

Here's the worst part: The geniuses at Pearson think they get it, that they're doing a great job with all this. They claim: "Pearson Realize™ is our newest learning management system that gives 'digital natives' the learning experience that they have come to expect."

> She thinks the struggle of working with bad software is necessary, a character builder, like hoeing potatoes in the Idaho sun.

Baloney. We don't expect badly structured, difficult-to-navigate software, though we encounter it far too often. It certainly isn't what we deserve.

I'm here to learn algebra, not to use your software. I shouldn't have to fight the program in order to do this. It should not take me longer to type in my answer than to calculate the actual problem. As Dave says, it's like doing a tonsillectomy through the rectum—pointlessly more painful.

I've already fought with my teacher over this point. She thinks the struggle of working with bad software is necessary, a character builder, like hoeing potatoes in the Idaho sun. She wouldn't get far with this attitude in Dave's UX class.

Bad software isn't "good for your character," it's a pain in the ass. There's a difference, and it's an important one. If more of us got mad about it, maybe we could change it. ■

ANNABELLE ROSE PLATT *is about to enter 11th grade in Ipswich, Mass. Her world greatly expanded at age 2, when she peeled the tape off her fingers and discovered the numbers 9 and 10. She does not agree with everything her father says.*

Looking for a powerful distributed cache?

Get the easiest, most powerful in-memory data grid designed to scale your .NET applications.

- Fast and linearly scalable
- Enterprise-grade availability
- Industry-leading ease of use
- Integrated in-memory computing

Go with the industry leader in distributed caching

Trusted by hundreds of enterprise customers for more than 12 years, ScaleOut's distributed caching technology delivers rock-solid performance, legendary ease of use, and world-class support. Unlike Redis, it offers a better migration path from AppFabric Caching. Here's why:

	ScaleOut	Redis
Source-code compatible AppFabric Caching APIs for seamless migration	✔	✖
Architected from the ground up for automatic scaling and high availability	✔	✖
Fully coherent data storage and access for mission-critical applications	✔	✖
Advanced .NET features: distributed LINQ query, C# MapReduce, and more	✔	✖

Replacing AppFabric Caching? Trouble switching to Redis?
Check out our AppFabric-compatible drop-in: **www.scaleoutsoftware.com/appfabric**

ScaleOut Software

Please visit **scaleoutsoftware.com** or call us at **+1.503.643.3422** to learn how ScaleOut's technology can help accelerate your business.

DevExpress®

Build Your Best
Without Limits or Compromise

WIN ASP WPF

What's in this Issue?

Official Supplier

As official and authorized distributors, we supply you with legitimate licenses directly from 300+ software publishers.

24/5 Customer Service

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

Trusted for Over 20 Years

Over 580,000 licenses delivered to Developers, SysAdmins, Corporations, Governments & Resellers, worldwide.

Open 24 hours a day, Monday to Friday

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

United States	888 850 9911
US Gov't Sales	888 850 9966
United Kingdom	0800 581111
Japan	0120 343 550
Argentina	0800 666 0185
Australia	1 800 0 15292
Austria	0800 281 750
Belgium	0800 7 5603
Brazil	0800 891 6478
Bulgaria	00800 115 4445
Canada	(888) 850 9911
Chile	1230 020 6857
China (North)	10800 481 1661
China (South)	10800 813 1594
Colombia	01800 710 2043
Croatia	0800 22255
Czech Republic	800 143 313
Denmark	80 88 17 20
Ecuador	1 800 010 562

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

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

US Headquarters

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

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

Languages:
English / Spanish / Portuguese

Office Hours:
9:00am - 7:00pm EST

European Headquarters

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

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

Languages:
English / French / German / Spanish / Italian

Office Hours:
9:00am - 5:30pm GMT & CET

Asia-Pac Headquarters

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

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

Languages:
English / Japanese / Korean / Mandarin / Cantonese

Office Hours:
10:00am - 6:00pm JST

/n software Red Carpet Subscription

Top 50 Product | Top 25 Publisher | Enterprise Partner

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

Publisher: /n software | Category: Communication & Messaging

Components for every major protocol from FTP to IMAP to SNMP, SSL and SSH security, S/MIME encryption, Digital Certificates, Credit Card Processing, ZIP compression, Shipping and Tracking, and e-business (EDI) transactions.

Encrypt/Decrypt Files and Emails

Easily implement strong encryption via major cryptographic standards. Features include encryption & decryption, secure message hashing, signing & signature verification and digital certificate generation.

Core Components

IP*Works! is a comprehensive framework for Internet development. It provides programmable, SSL-enabled components for sending email, transferring files, managing networks, browsing the Web and much more.

Internet E-Commerce

Add secure and reliable Internet payment processing to your applications. Includes components for Credit Card and Electronic Check (ACH) processing via major Internet payment gateways.

/n software

Prices from $ 1,535.04

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

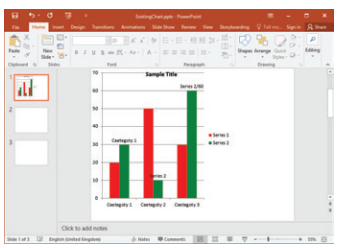
Aspose.Total for .NET

#5 Product | #3 Publisher | Enterprise Partner

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

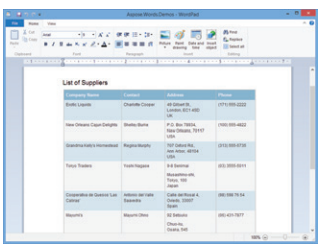
Publisher: Aspose | Category: Document & Text Processing | ★★★★★

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



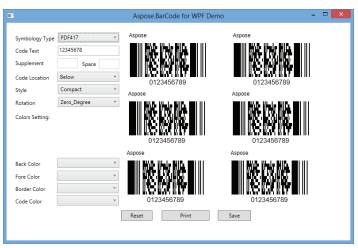
Aspose.Slides for .NET

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



Aspose.Words for .NET

An advanced class library that lets you perform a wide range of document processing tasks directly within your applications. Supports DOC, OOXML, RTF, HTML, OpenDocument, PDF, XPS, EPUB and many other formats.



Aspose.BarCode for .NET

Allows developers to quickly and easily add robust and reliable barcode generation and recognition functionality to their Microsoft .NET applications. It supports WinForms and ASP.NET.

ASPOSE File Format APIs

Prices from $ 2,939.02

www.componentsource.com/aspose-total-net

Aspose.Pdf for .NET

Top 25 Product | #3 Publisher | Enterprise Partner

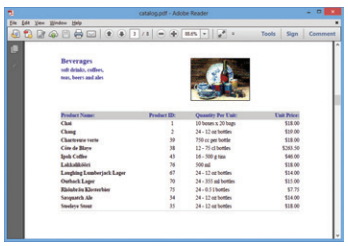
Create and manipulate PDF documents without using Adobe Acrobat.

Publisher: Aspose | Category: PDF | ★★★★★

Aspose.Pdf for .NET is a set of PDF APIs that enables your .NET applications to read, write and manipulate existing PDF documents. It also allows you to create forms and manage form fields embedded in a PDF document.

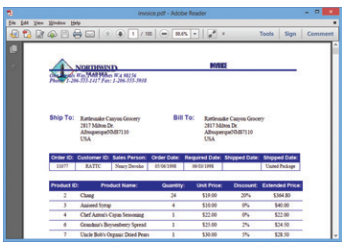
Table Creation

Access, create and modify tables, rows and cells. You can also add headings and Table of Contents during PDF creation and programmatically format document elements.



Document Manipulation

Concatenate or merge two or more PDF documents, append new pages to an existing PDF file and extract or insert pages. Document formatting is supported using an advanced document object model.



Convert PDFs

Convert between PDF and other popular document formats including HTML, PCL, DOCX, XLS, SVG, JPG, PNG, TIFF and XPS. You can also transform XML and XSL-FO documents into PDFs.

ASPOSE File Format APIs

Prices from $ 979.02

www.componentsource.com/aspose-pdf-net

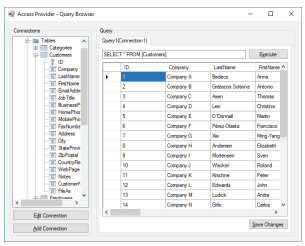
CData ADO.NET Subscription

Comprehensive access to data from ADO.NET.

Top 50 Publisher | Enterprise Partner

Publisher: CData Software | Category: Data Access

Real-time data integration from 80+ SaaS, NoSQL, and Big Data sources. Includes a single SQL interface to data that insulates users from the challenges and complexities of integrating with individual APIs, SDKs, and services.

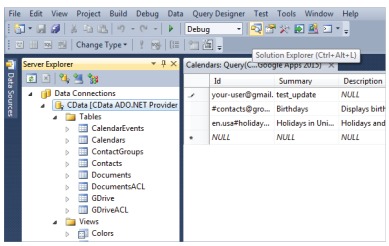


Access Data from any .NET App

Use objects to connect to data just as you would any traditional database, through the Visual Studio Server Explorer, in code, and in data controls like DataGridView, GridView and DataSet.

Connect .NET Apps to Data

Comprehensive access to application, database, and Web API data through easy-to-use tools. Whether you are building Desktop, Web, or Mobile apps you can use the drivers for fast and secure access to data.



Visual Studio Integration

The ADO.NET Data Providers can be used to access and explore data directly from the Visual Studio Server Explorer. These constructs return live data that you can work with directly from within Visual Studio.

Prices from $ 979.02

www.componentsource.com/cdata-ado-net-subscription

CData Excel Add-In Subscription

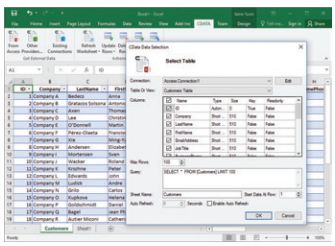
Top 50 Publisher

Enterprise Partner

Access to applications, databases, and Web APIs directly from Excel worksheets.

Publisher: CData Software | Category: Data Access Infrastructure

CData Excel Add-In Subscription gives Excel users unlimited access to real-time data from an extensive list of applications, databases, and SaaS sources. From Big Data and NoSQL databases, to CRM, ERP, and accounting systems.

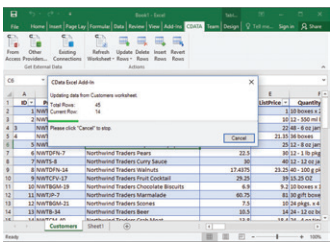


Easily Connect to Data

Users simply supply their credentials to create a connection and can immediately begin working with live data. The Excel Add-In is completely self-contained; no additional software installation is required.

Included Data Sources

CData Excel Add-In Subscription includes 75+ data sources including Amazon DynamoDB, Authorize.Net, Facebook, Google AdWords, Microsoft Access, MySQL, Office 365, PayPal, QuickBooks and Xero Accounting.



Bi-Directional Access

Perfect for mass imports/exports/updates, data cleansing and Excel based data analysis. You can also modify and delete records, quickly export and backup data, and work on data with charts and pivot tables.

cdata

Prices from $ 293.02

www.componentsource.com/cdata-excel-add-in-subscription

CData ODBC Driver Subscription

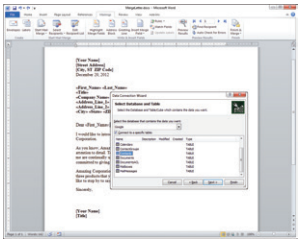
Top 50 Publisher

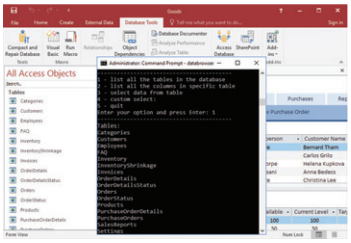
Enterprise Partner

Connecting to data has never been easier.

Publisher: CData Software | Category: Data Access

CData ODBC Driver Subscription gives you access to live SaaS Apps, NoSQL, and Big Data from desktop clients through standard ODBC connectivity.





Read & Write Data through ODBC

CData ODBC Driver Subscription includes 75+ data sources including Apache HBase, Azure Table, Gmail, Google Sheets, MailChimp, Microsoft Dynamics, SharePoint, SQLite, Sugar CRM, and YouTube Analytics.

Connect from Office Tools

Any application that can access data through ODBC, like Microsoft Excel, PowerPivot, or Word, can leverage the ODBC Drivers to connect to real-time data and help you with powerful capabilities like mail merge.

Connect to Live Data

The drivers include a library of 50+ functions, including String, Date and Numeric that can manipulate column values into the desired result. Popular examples include Regex, JSON, and XML processing functions.

cdata

Prices from $ 391.02

www.componentsource.com/cdata-odbc-driver-subscription

CData PowerShell Cmdlet Subscription

Access data directly from the PowerShell command-line.

Publisher: CData Software | Category: Data Access Infrastructure

The PowerShell Cmdlet Subscription offers straightforward access to live data from popular NoSQL and Big Data databases, CRM, ERP, accounting systems, marketing automation, cloud platforms, and more.

PowerShell Data Integration

Comprehensive access to application, database, and Web API data through familiar tools. The included Cmdlets enable PowerShell users to quickly and easily work with live data from virtually anywhere.

Clean & Normalize Data

Use PowerShell scripting to normalize and/or de-duplicate data or pipe data from various sources into flat-files, databases, and other data stores for archival, back-up and synchronization purposes.

Real-time Access to Data

The Cmdlets offer a simple Transact-SQL interface for connecting with data, in a virtually identical way to how a user would interact with a relational database from PowerShell.

Prices from $ 391.02

www.componentsource.com/cdata-powershell-cmdlet-subscription

CData SSIS Component Subscription

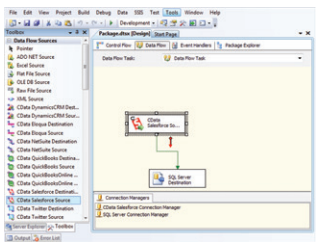
Top 50 Publisher

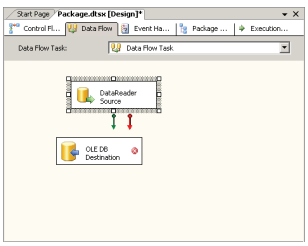
Enterprise Partner

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

Publisher: CData Software | Category: Data Access Infrastructure

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





SQL Server Integration

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

Connect SQL Server with Data

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

Included Data Sources

Integration with 80+ SaaS, NoSQL, & Big Data sources including Amazon SimpleDB, FreshBooks Accounting, Google Apps, Microsoft Dynamics, Office 365, QuickBooks, Salesforce & Force.com and Sugar CRM.

cdata

Prices from $ 783.02

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

Know SQL?

Connect to 80+ data sources (NoSQL, Big Data, SaaS, etc.) through standard SQL queries.

ODBC ADO.NET SSIS CLOUD BIZTALK EXCEL MOBILE

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

Learn more: www.componentsource.com/cdata

cdata

DevExpress Universal

#2 Product | #1 Publisher | Enterprise Partner

All DevExpress Visual Studio products in one integrated subscription.

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

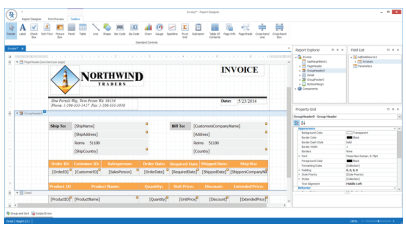
Includes charts, data grids, spreadsheets, calendars, schedulers, diagrams, navigation, text editors, PDF, maps, gauges, tiles and data editors. Supports ASP.NET WebForms & MVC, WinForms, WPF, Windows 10 and HTML5/JS.

HTML5/JS Framework

A JavaScript development framework that allows you to create store-ready, multi-device applications across platforms and devices. Each widget includes a jQuery plugin and bindings to AngularJS and KnockoutJS.

Dashboard

Create insightful and information rich decision support systems for executives and business users across platforms and devices. Results are immediate, accurate and always relevant.



Reporting for .NET

DevExpress Reporting delivers easy-to-use customization options and a rich set of report controls, including multi-dimensional pivot tables & charts so you can build reports of unmatched informational clarity.

DevExpress™

Prices from $ 2,111.99

www.componentsource.com/devexpress-universal

DevExpress DXperience

#1 Product | #1 Publisher | Enterprise Partner

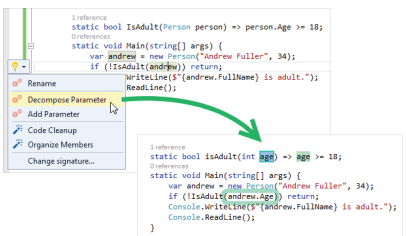
All DevExpress .NET Framework control libraries in one integrated subscription.

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

DevExpress DXperience helps you build applications for Windows, Web, mobile and tablet with WinForms, ASP.NET, WPF and Windows 10 controls. Source code is included for all controls and libraries that ship as part of the Subscription.

Windows 10 App Controls

Includes 20+ elegant, touch-first UWP controls for Windows 10 solutions. They allow you to create compelling business solutions that emulate the look, feel and capabilities of Microsoft Office.



CodeRush for Roslyn

Write code at the speed of thought, find and fix defects and Refactor with ease. CodeRush makes it easier to see what's going on with complex code, so you can move forward and spend less time wondering.

ASP.NET Rich Text Floating Objects

Freely position, scale and rotate floating objects. Whether used for images or as a text box, users can modify the object's characteristics via an integrated context (popup) menu or a Ribbon context tab.

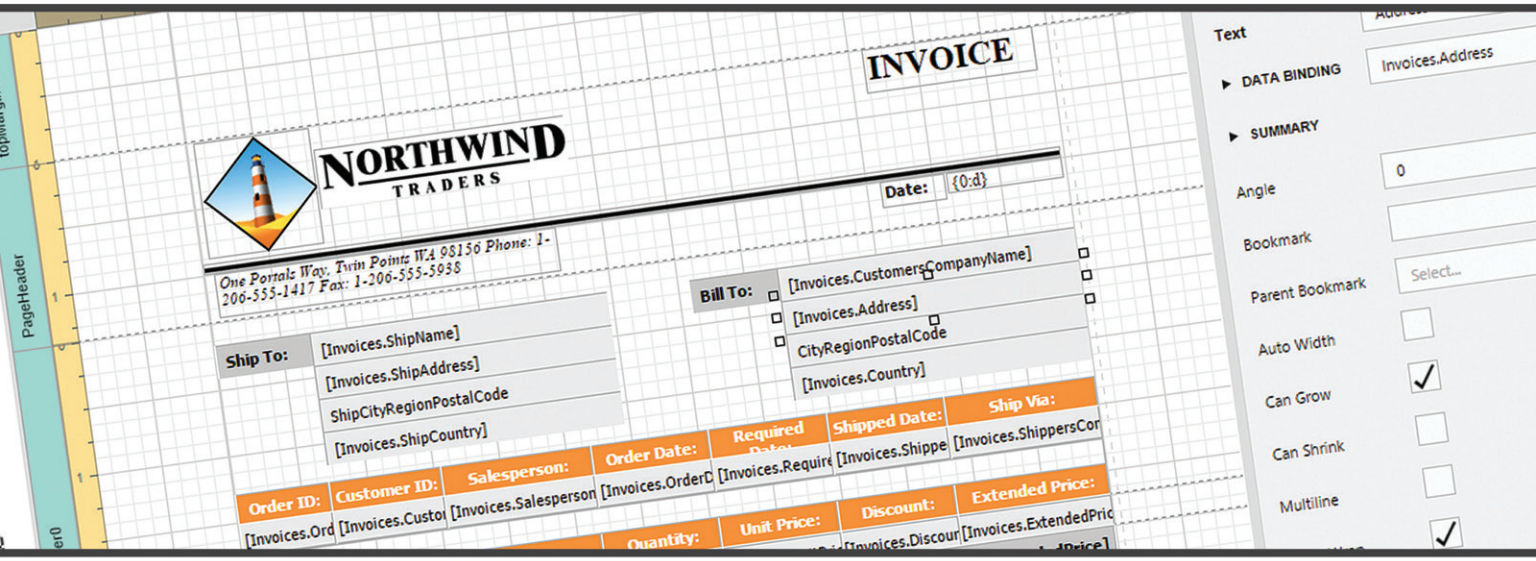
DevExpress™

Prices from $ 1,439.99

www.componentsource.com/devexpress-dxperience

Reporting Made Easy

Inform and analyze with absolute ease. DevExpress Reporting includes an enterprise-ready reporting platform with integrated Windows & Web report designers.



Your Next Great Business Report Starts Here

See the power and flexibility of our reporting platform in action.
componentsource.com/devexpress-reporting

DevExpress®

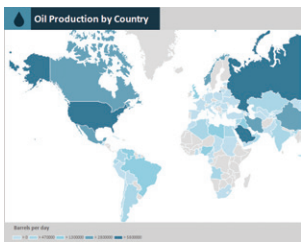
ActiveReports

#3 Product | #2 Publisher | Enterprise Partner

A flexible and extensible platform for every kind of reporting.

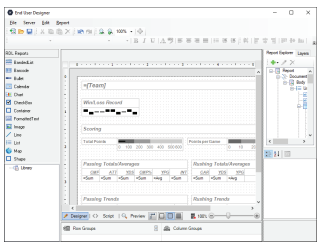
Publisher: GrapeCity | Category: Reporting

Design, customize, publish, and view data in your business applications. Create everything from simple invoices to complex statistical data visualizations, and you can even enable your end users to design reports for themselves.



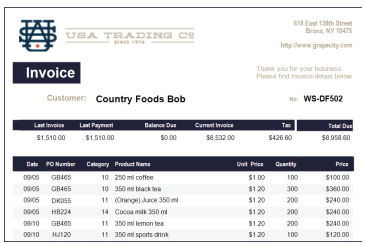
.NET Reporting Components

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



Customizable End-User Designer

Embed a report design environment in your own WinForms applications, giving end users the power to design their own reports in a code-free ad-hoc reporting environment.



Multi-Platform Report Viewers

Embed the fully customizable viewers in your HTML5, WPF, WinForms or ASP.NET MVC applications. PDF, Excel, Word and other export formats let you open report output in other programs.

GrapeCity

Prices from $ 1,575.02

www.componentsource.com/activereports-developer

A comprehensive .NET reporting platform for every application

Design, publish, view, print, and export all of your reports. Create operational reports, such as tax and government forms, invoices, and expenses. Sales performance, budgeting, and revenue analysis is easier than ever with built-in strategic and analytical reports. ActiveReports gives you the power and flexibility you need to turn your data into informative, pixel-perfect reports across the enterprise.

1996
20+ years on the market

300,000+ developer community

Multi-platform report viewers

Trusted by a worldwide customer base

Fast reporting

LEARN MORE AT WWW.COMPONENTSOURCE.COM/ACTIVEREPORTS

GrapeCity ActiveReports

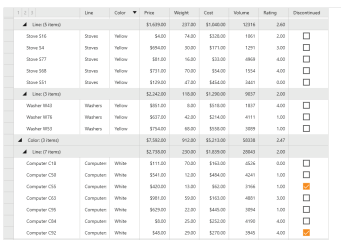
ComponentOne Studio

Top 25 Product | #2 Publisher | Enterprise Partner

A complete collection of award-winning .NET UI controls for mobile, web and desktop.

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

Includes grids, data management, data visualization, reporting, documents, scheduling, input, editing, navigation, layout and utilities. Supports WinForms, WPF, UWP, ASP.NET MVC, ActiveX, LightSwitch, Silverlight, ASP.NET Web Forms.

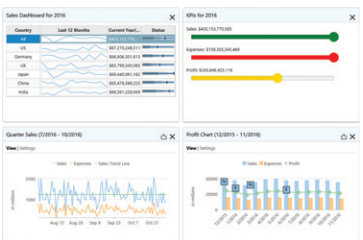


Industry Leading Grid

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

Engaging User Interactions

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



Powerful Data Visualization

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

GrapeCity

Prices from $ 1,472.58

www.componentsource.com/componentone-studio

SpreadJS

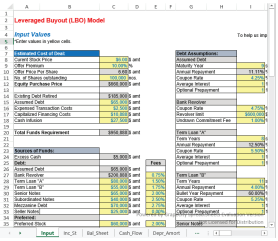
#2 Publisher

Enterprise Partner

Enterprise JavaScript spreadsheet and data presentation components.

Publisher: GrapeCity | Category: Spreadsheet

SpreadJS includes powerful JavaScript spreadsheet and grid components that developers can use to create data views, numerical models, forms, dashboards, and reports for the Web.

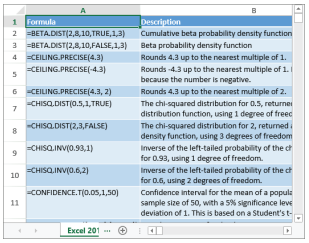


JavaScript Spreadsheet

Display and manage data much like Microsoft Excel with features such as formula engine, sorting, filtering, input controls, sparklines, and native Excel file import and export. SpreadJS is optimized for speed.

Data Display and Visualization

SpreadJS includes sparklines that allow you to quickly and professionally summarize data trends. You can also show performance indicators, data bars, color scales, and more with conditional formatting support.



Function & Calculation Engine

Choose from over 300 built-in functions and apply them to individual cells, rows/columns, or entire sheets. You can also easily add an Excel-like formula text box to your sheets.

GrapeCity

Prices from $ 984.02

www.componentsource.com/spreadjs

Wijmo Enterprise

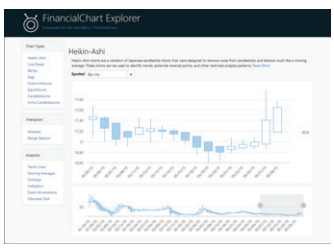
#2 Publisher

Enterprise Partner

Lightning-fast, touch-first, flexible controls for your enterprise applications.

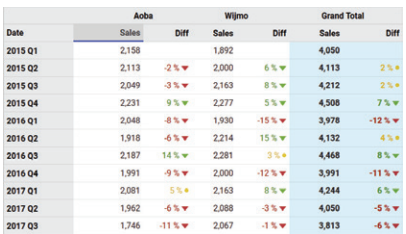
Publisher: GrapeCity | Category: Presentation Layer

Wijmo's JavaScript/HTML5 controls are written in TypeScript for enterprise application development. With touch-first design and full Angular support, Wijmo's FlexGrid and chart controls focus on top performance and zero dependencies.



FinancialChart for JavaScript

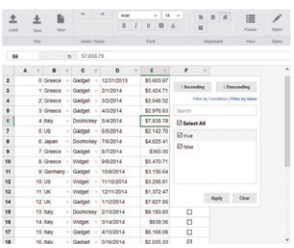
Instantly create stunning, advanced stock trending visualizations with the powerful FinancialChart for Wijmo. Analyze with trendlines, filters, range selectors and annotations - all with minimal coding.



	Aoba		Wijmo		Grand Total	
Date	Sales	Diff	Sales	Diff	Sales	Diff
2015 Q1	2,158		1,892		4,050	
2015 Q2	2,113	-2 %	2,000	6 %	4,113	2 %
2015 Q3	2,049	-3 %	2,163	8 %	4,212	2 %
2015 Q4	2,231	9 %	2,277	5 %	4,508	7 %
2016 Q1	2,048	-8 %	1,930	-15 %	3,978	-12 %
2016 Q2	1,918	-6 %	2,214	15 %	4,132	4 %
2016 Q3	2,187	14 %	2,281	3 %	4,468	8 %
2016 Q4	1,991	-9 %	2,000	-12 %	3,991	-11 %
2017 Q1	2,081	5 %	2,163	8 %	4,244	6 %
2017 Q2	1,962	-6 %	2,088	-3 %	4,050	-5 %
2017 Q3	1,746	-11 %	2,067	-1 %	3,813	-6 %

OLAP Pivot Controls

Create Excel-like, web-based pivot tables in JavaScript with Wijmo's OLAP module. Process and aggregate thousands of rows in milliseconds with no server-side dependencies.



JavaScript Spreadsheet

Produce an Excel-like experience in your web app with a fast, lightweight control allowing users to import spreadsheets, apply formulas, format and freeze cells, undo/redo, and export data to Excel.

GrapeCity

Prices from $ 1,177.08

www.componentsource.com/wijmo

ComponentOne Studio for Xamarin

#2 Publisher

Enterprise Partner

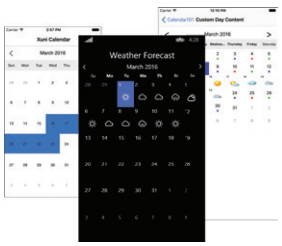
Code once in C# and XAML across native mobile devices.

Publisher: GrapeCity | Category: Data Visualization

ComponentOne Studio for Xamarin enables you to deliver beautiful cross-platform native mobile apps with flexible controls including, grids, charts and gauges with adaptive styles for any device. Target Windows 10 apps as easily as you do iOS & Android.

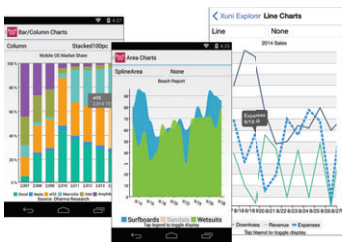
FlexGrid

Display tabular data across columns and rows with this powerful grid control. FlexGrid for Xamarin brings a spreadsheet-like experience to your apps, with intuitive touch gestures and quick-cell editing capabilities.



Calendar

Calendar for Xamarin comes complete with intuitive navigation and selection gestures, animation and built-in globalization. Customize the appearance with horizontal or vertical navigation and custom display settings.



FlexChart

Visualize your data in a wide range of charts, including bar, column, area, line, spline, scatter, bubble & candle. With animation and interactive features, FlexChart for Xamarin brings life to your mobile dashboards.

GrapeCity

Prices from $ 881.58

www.componentsource.com/componentone-studio-for-xamarin

Infragistics Ultimate

Top 25 Product | #4 Publisher | Enterprise Partner

UX & UI tools built to accelerate the application design & development process.

Publisher: Infragistics | Category: Presentation Layer

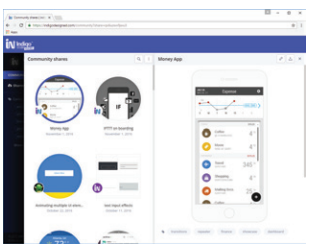
Hundreds of UI controls including charts, grids and MS Office-compatible frameworks. Supports ASP.NET & MVC, JavaScript, Windows Forms, WPF, Xamarin, Android and iOS. Also includes prototyping, usability testing and data visualization tools.

Native Mobile Controls

Includes everything you need to build native mobile apps with dynamic charts and high-performance data grids. Using Infragistics Xamarin.Forms you can architect multiple experiences with a single codebase.

Build Responsive Web Applications

Works with Angular, React and other popular frameworks. Create modern Web applications with fast, easy-to-use UI components and productivity tools. Use Ignite UI to create applications natively in Angular.



Code-free Prototyping

Indigo Studio empowers designers, developers, product managers and users to quickly and collaboratively create and test realistic prototypes - before the first line of code is written.

INFRAGISTICS

Prices from $ 1,955.10

www.componentsource.com/infragistics-ultimate

Infragistics Ignite UI

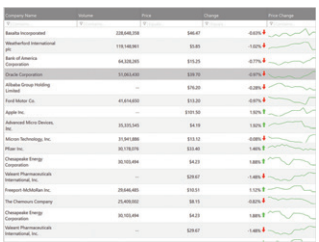
#4 Publisher

Enterprise Partner

Create high-performance, touch-first, responsive apps.

Publisher: Infragistics | Category: Presentation Layer

A complete HTML5 & JavaScript toolkit to build modern browser experiences on any device - desktop, tablet or phone. Create powerful Angular and React modern Web applications with fast, easy-to-use UI components and productivity tools.



Build Modern Apps

Ignite UI plays nice with all your favorite app frameworks, data binding libraries and IDEs. With full support for AngularJS, Angular, Knockout JS, ASP.NET MVC, Bootstrap, React and more, the choice is yours.

Lightweight and Fast

Ignite UI delivers best-in-class performance for your high-demand data needs. The virtualized, load-on-demand data grids can handle any data scenario.

Data Visualizations

With HTML5 charts, gauges, financial charts, statistical and technical indicators, trend lines and more, deliver the most demanding and beautiful touch-friendly dashboards for desktop and mobile apps.

INFRAGISTICS

Prices from $ 681.10

www.componentsource.com/infragistics-ignite-ui

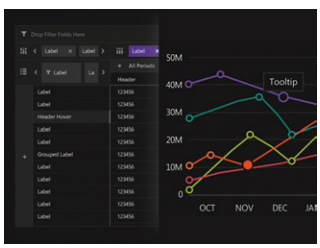
Infragistics Professional

Top 10 Product | #4 Publisher | Enterprise Partner

Build functional and stylish Enterprise-ready applications.

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

Get performance, ease of use, and styling with a powerful suite of UI components that allows you to develop across all browsers, devices, and platforms. Contains hundreds of UI controls, including grids, charts, maps and MS Office-compatible frameworks.



Touch-friendly WPF Controls

Bring modern, engaging apps for desktop and touch-screen devices to market while shortening development time. Includes fast data grids, charts, dynamic data visualization, scheduling, styling and themes.

JavaScript/HTML5 & ASP.NET MVC

Build high-performing, responsive Web applications in JavaScript/HTML5 for Angular and React frameworks. Get a jump-start on the most demanding Web applications with ASP.NET MVC server-side wrappers.

Time-Saving Xamarin Controls

45+ native controls for Xamarin.iOS, Xamarin.Android, and Xamarin.Forms enable maximum code-sharing and incredible performance. Includes grids, charts, and exclusive productivity tools.

INFRAGISTICS

Prices from $ 1,465.10

www.componentsource.com/infragistics-professional

Infragistics Ultimate UI for Xamarin

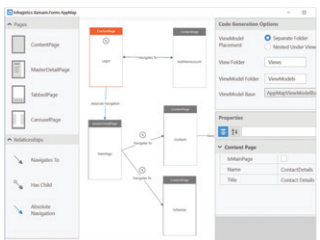
Create fast, beautiful, cross-platform mobile apps.

#4 Publisher

Enterprise Partner

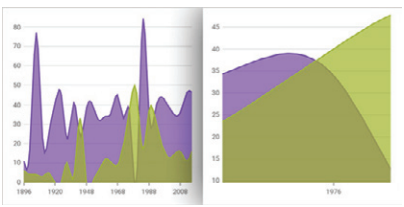
Publisher: Infragistics | Category: Presentation Layer

Includes 45+ native controls for Xamarin.iOS, Xamarin.Android, and Xamarin.Forms that enable code-sharing and incredible performance. Includes grids, business charts (bar, column, pie, area), financial and sparkline charts, gauges and more.



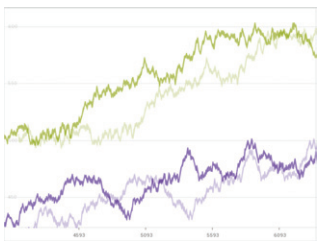
Time-Saving Productivity Pack

Includes Xamarin Productivity Pack with control configurators, AppMap, and NuGet-powered Xamarin.Forms Toolbox. Streamline your coding so you can focus on innovating and solving business problems.



Animated Transitions

Choose from 18 different animated transitions based on the movement and direction of animated data points. Transitions include accordion, expansion from value midpoints, zero value, sweep from side and more.



Fast Performance

Rapid data visualization delivers powerful performance with even the largest real-time streaming datasets. Visualize hundreds of thousands of data points, without sacrificing quality or frame rate.

INFRAGISTICS

Prices from $ 783.02

www.componentsource.com/infragistics-ultimate-ui-for-xamarin

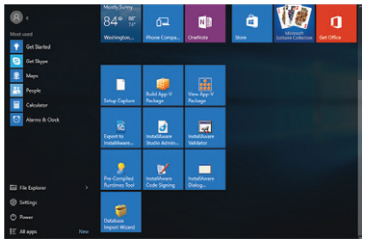
InstallAware Studio Admin

Top 100 Product | Top 50 Publisher | Enterprise Partner

A powerful software installation solution with a Web-based automation interface.

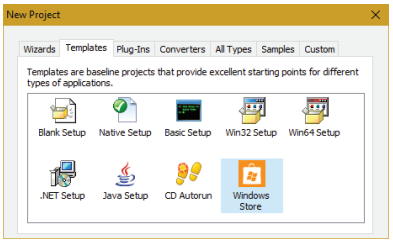
Publisher: InstallAware | Category: Release Automation & Management

InstallAware Studio Admin is a software installation solution for Windows that provides you with the advantages of true Windows Installer rapid development productivity.



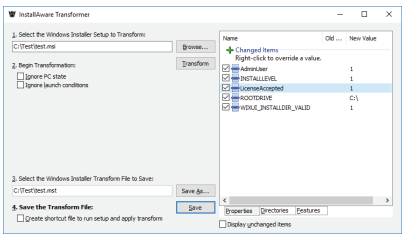
Latest Technology Support

InstallAware supports Windows 10 and Visual Studio 2017. Display installation progress on the Windows Taskbar, and pin apps to the new Windows 10 Start Menu Live Tiles.



Windows Store Bridge

InstallAware effortlessly bridges your apps to the Windows Store, creating a Universal Windows app from a customizable template and helping your end-users download your apps directly from the Windows Store.



InstallAware Transformer

Give non-developers a point-and-click way to create MST transform files for any MSI setup. Any user may run through the setup, and instead of installing the package, InstallAware creates an MST transform file.

InstallAware

Prices from $ 3,919.02

www.componentsource.com/installaware-studio-admin

InstallAware Studio

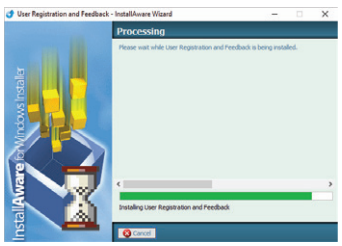
Top 50 Publisher

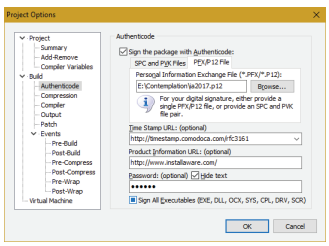
Enterprise Partner

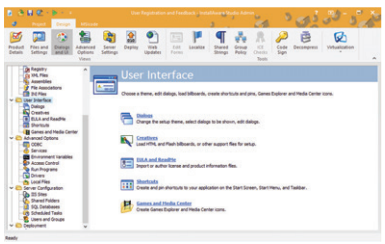
A powerful software installation solution that supports the latest technologies.

Publisher: InstallAware | Category: Release Automation & Management

InstallAware Studio is a software installation solution for Windows that provides you with the latest features and technology support, with the advantages of true Windows Installer rapid development productivity.







Faster, More Reliable Installs

Ships with a Native Code Setup Engine, a faster and more reliable alternative to Windows Installer. The Native Engine with InstantInstall Acceleration creates Windows Installers faster by an order of magnitude.

SHA 256 Code Signing

InstallAware applies a smart algorithm to determine whether files are going to be singly signed, doubly signed, or skipped. All signing happens automatically to comply with Microsoft's code signing standards.

Easy MSI Customization

InstallAware setups don't have any dependencies or require scripting runtimes. It is possible to completely customize your setup even if you're unfamiliar with Windows Installer.

InstallAware

Prices from $ 1,959.02

www.componentsource.com/installaware-studio

Syncfusion Essential Studio Enterprise

Top 100 Product | Top 50 Publisher | Enterprise Partner

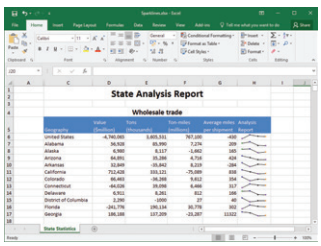
Deliver better solutions with a complete suite of .NET & JavaScript controls.

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

Includes 800+ components across Windows Forms, WPF, ASP.NET MVC, ASP.NET Web Forms, LightSwitch, Silverlight, iOS, Android, Windows Phone, Xamarin, JavaScript and UWP. No royalties, run-time, or server deployment fees.

ASP.NET Web Forms

A comprehensive suite of ASP.NET Web Forms components for Enterprise Web development. It includes several complex widgets such as data grid, chart, gantt, diagram, spreadsheet, schedule, pivot grid and much more.



Read and Write Excel Files

Enable any .NET application to create, read, write and process Microsoft Excel files. Features access to each aspect of the file, blazing performance, no server deployment fees and easy migration from Office Automation.

Mobile Development

Essential Studio Enterprise comes with over 50 controls for developing Xamarin.Forms, Xamarin.iOS, and Xamarin.Android applications. It includes Excel, Word and PDF manipulation controls, data grids and charts.

Syncfusion

Prices from $ 1,795.50

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

Syncfusion Essential Studio for Xamarin

Top 50 Publisher

Enterprise Partner

Comprehensive component suite for Xamarin.iOS, Xamarin.Android & Xamarin.Forms.

Publisher: Syncfusion | Category: Presentation Layer

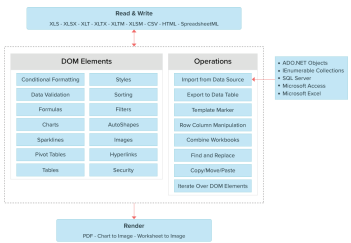
50+ components including charts, gauges, datagrid, treemap, Excel, Word, PDF, data input, autocomplete and barcode. Easily integrate the components within your applications, with thorough documentation, knowledge base articles and samples.

Plot 20+ Chart Types

The Chart control can plot over 20 chart types, from line charts to specialized financial charts. Its rich feature set includes functionalities like data binding, multiple axes, trackball, tooltip and zooming.

Essential DataGrid for Xamarin

A grid component built to achieve the best performance on the Xamarin platform. It offers responsive touch, scrolling 100,000s of records and includes advanced features such as grouping, sorting and filtering.



Read/Write Excel Files

Essential XlsIO is a library that can read and write Microsoft Excel files. It includes a comprehensive API, has no external dependencies and can be used on systems without Excel installed.

Syncfusion

Prices from $ 975.10

www.componentsource.com/syncfusion-essential-studio-for-xamarin

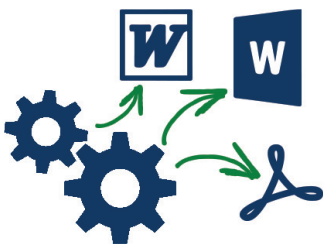
TX Text Control .NET Server for ASP.NET

Top 100 Product | Top 25 Publisher | Enterprise Partner

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

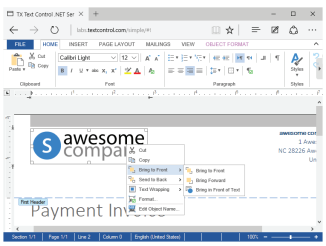
Publisher: Text Control | Category: Word Processing

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



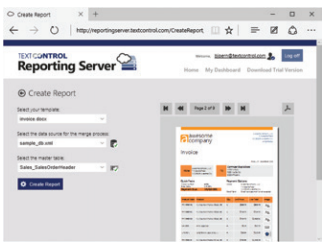
Document Conversion

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



Cross-Browser Document Editing

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



Automate Your Word Documents

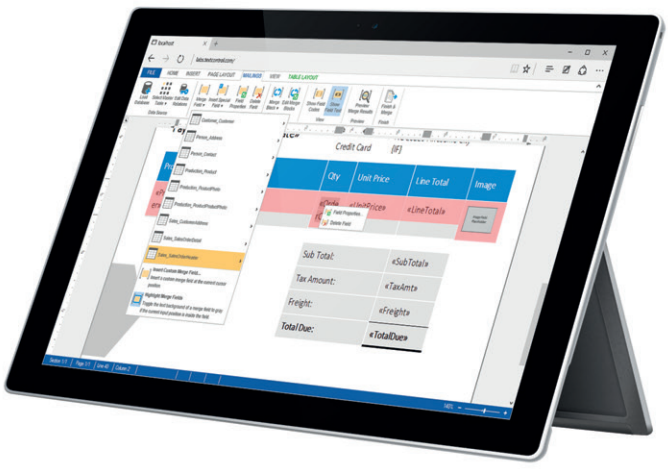
Using TX Text Control you can automate, edit and create documents using UI and non-UI components, all without Microsoft Office Automation. It also gives you access to flow type layout reporting features.

TEXT CONTROL

Prices from $ 2,938.04

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

ASP.NET MVC REPORTING



The first cross-browser, true WYSIWYG HTML5-based document editor.

Now with full ASP.NET MVC support.

Edit and create MS Word documents

Create and modify Adobe® PDF documents

Create reports and mail merge templates

Integrate with Microsoft® Visual Studio

```
PM> Install-Package TXTextControl.Web
```

Give your users an MS Word compatible editor to create powerful documents and reporting templates anywhere - in any browser. Feature-complete including mail merge, sections, headers and footers, drawings and shapes, tables, barcodes and charts.

Available for ASP.NET Web Forms and MVC.

Find out more about TextControl:
www.componentsource.com/textcontrol

Visual Studio Partner

Software • Training • Consulting

TEXT CONTROL

TX Text Control .NET for Windows Forms

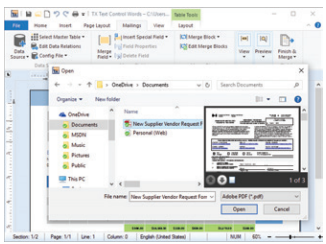
Top 25 Publisher

Enterprise Partner

Add Microsoft Word look and feel editing to your Windows Forms applications.

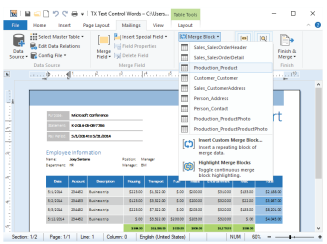
Publisher: Text Control | Category: Word Processing

TX Text Control .NET for Windows Forms is a royalty-free, fully programmable rich edit control that offers developers a broad range of word processing features in a reusable component for Visual Studio.



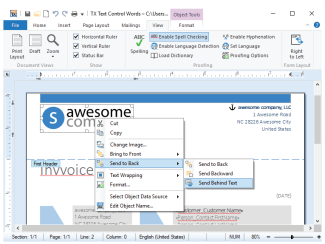
Import Adobe PDF Documents

Import PDF documents, and view, edit or convert them. Imported PDFs can be modified just like any other format. The fully featured API can be used to change the content or to search on the document.



Reporting and Mail Merge

Text Control combines the power of a programmable reporting tool and WYSIWYG word processor. Create mail merge and table reports, master-detail and sub-reports based on MS Word compatible report templates.



Word Compatible File Formats

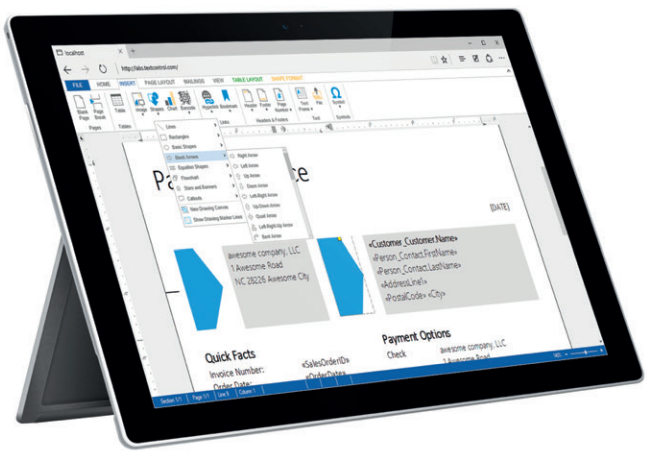
Modify MS Word documents or create Adobe PDF documents from your application. Supports a range of word processing formats (RTF, DOC, DOCX, HTML, XML, PDF) and image formats (GIF, PNG, JPG, BMP, WMF, EMF, TIF).

TEXT CONTROL

Prices from $ 1,045.59

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

ALL ABOUT DOCUMENTS



Text Control provides deep functionality components and consulting to integrate reporting and word processing into .NET Windows, Web and Mobile applications.

Visual Studio Partner

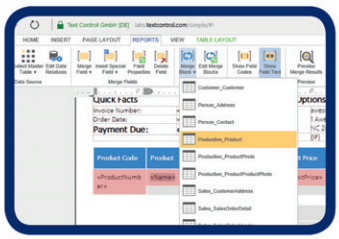
HTML

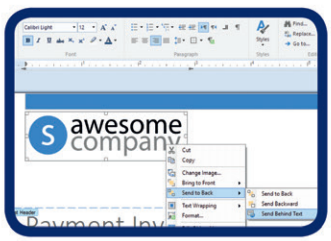
Software • Training • Consulting

Reporting and Mail Merge

The Text Control Reporting Framework combines powerful reporting features with an easy-to-use, MS Word compatible word processor for ASP.NET, WPF and Windows Forms. Users can create documents and templates using ordinary Microsoft Word skills.

TX Text Control is completely independent from MS Word or any other third-party application and can be completely integrated into your business application.





MS Word compatible Rich Text Editing

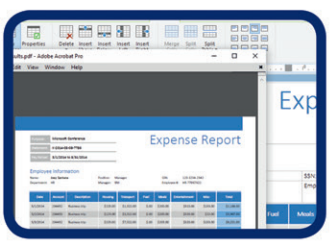
These feature-complete, fully programmable rich edit controls offer developers a broad range of word processing features in reusable UI and non-UI components for ASP.NET, WPF and Windows Forms.

They provide comprehensive text formatting, powerful mail merge features and all word processing key concepts such as table support, images, headers and footers, page sections and spell checking.

Load and Save Adobe PDFs

Using TX Text Control, Adobe PDF and PDF/A documents can be created including document access security settings and digital signatures.

PDF documents can be imported to be viewed, edited and converted. Edit PDF documents just like any other format such as DOC or DOCX. A fully featured API can be used to modify the content or to search through the document.



Find out more about TextControl:

www.componentsource.com/textcontrol

TEXT CONTROL

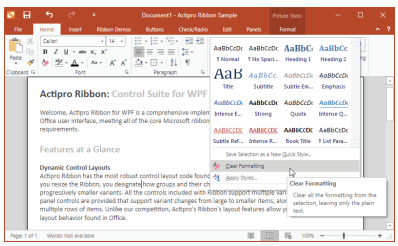
Actipro WPF Studio

Top 100 Product | Top 50 Publisher | Corporate Partner

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

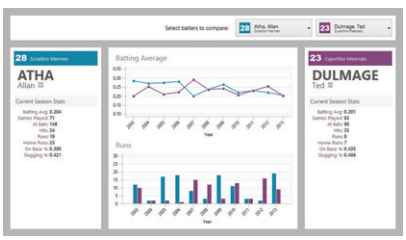
Publisher: Actipro Software | Category: Presentation Layer | ★★★★★

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



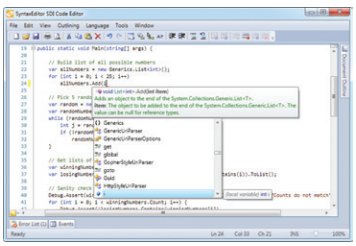
Ribbon Control

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



Visualize Complex Data

Actipro Charts is a set of full-sized charts that provide rich visualization for quantitative data. It's designed to make even the most complex data easily readable.



Code Editing in Your Apps

SyntaxEditor is a powerful text editing control that is packed with features for efficient code editing, including syntax highlighting, code outlining, parsing, line numbers and block selection.

actipro

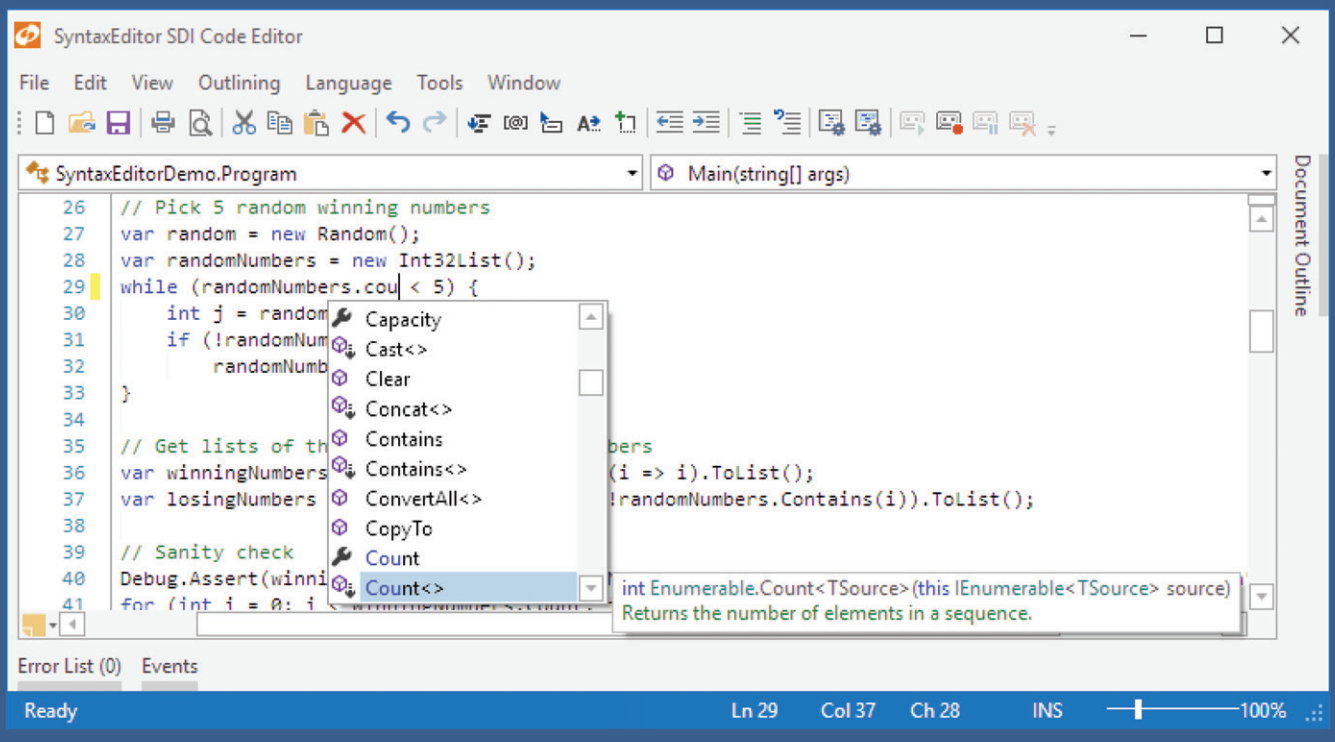
Prices from $ 636.02

www.componentsource.com/actipro-wpf-studio

Actipro SyntaxEditor

for WPF, Universal Windows, Silverlight, WinForms

The ultimate syntax-highlighting code editor control



Discover the possibilities...

componentsource.com/actipro

actipro

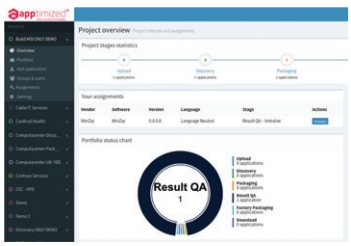
Apptimized

Corporate Partner

Cloud-based application packaging, start packaging for Windows 10 now.

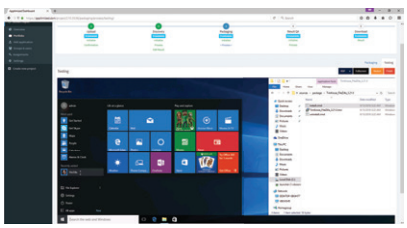
Publisher: Apptimized | Category: Cloud Services

Test, remediate, package and manage unlimited desktop applications for one low fee. Apptimized is completely cloud based, no need to stand up any hardware or software, and ready to support Windows 10 today.



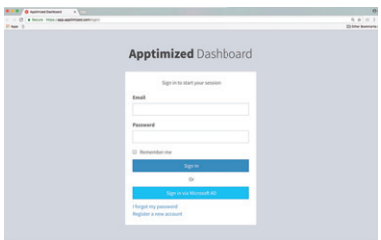
Unlimited Apps - One Low Fee

Everything needed to test, remediate, manage and create unlimited customized application packages for one low fee.



Package for Windows 10 Today

Apptimized can be set-up and configured to deliver customized MSI and App-V packages to individual requirements within minutes. Start packaging for Windows 10 now.



Package Anywhere, Anytime

Apptimized's unique cloud-based solution means that packages can be created without the need for any hardware or software, just an internet connection. Package anywhere, anytime, on any device.

apptimized® APPLICATION PACKAGING

Prices from $ 5,292.00

www.componentsource.com/apptimized

Janus WinForms Controls Suite

Top 50 Product | Top 25 Publisher | Corporate Partner

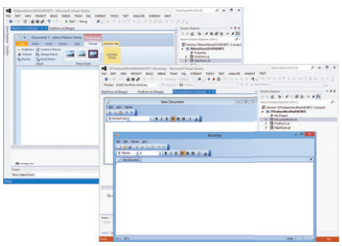
Add Outlook style interfaces to your .NET applications.

Publisher: Janus Systems | Category: Presentation Layer | ★★★★½

Janus WinForms Controls Suite is a set of controls designed to provide powerful user interfaces for Microsoft .NET Smart Client applications. All controls are 100% C# managed controls, designed to be used with Visual Studio.

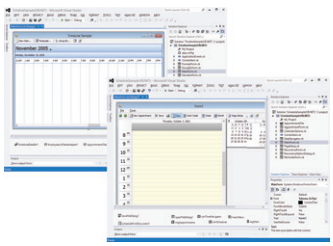
Janus GridEX for .NET

Create data grids with a UI similar to Microsoft Outlook, while easing development and maintenance time. Display, format, edit, sort, group, filter, manipulate, summarize, preview and print your data.



Janus UI Controls for .NET

Janus UI Controls features a range of user interface components including docking, toolbars, tabs, explorer, ribbons & buttons. Easily create SDI or MDI tabbed windows with auto hide and floating functionality.



Schedule & Timeline Controls

Present appointments in views, representing Days, WorkWeeks, Weeks, and Months. Easily load, display, format, group, filter, edit, and manipulate items in both bound and unbound modes.

janus systems

Prices from $ 889.00

www.componentsource.com/janus-winforms-controls-suite

Janus WinForms Controls Suite v4

Add powerful Outlook style interfaces to your applications

Janus WinForms Controls Suite enhances your applications with a powerful user interface that features the familiar Outlook visual style. All controls are 100% .NET managed code, 64-bit compatible and support Visual Studio and .NET Framework Client Profiles.

Janus WinForms Controls Suite includes sample applications that let you quickly add Outlook look and feel to your applications.

Learn more at:
www.componentsource.com/janus-systems

LEADTOOLS Multimedia SDK

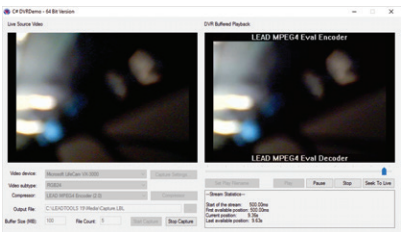
Top 25 Publisher

Corporate Partner

Create professional, high-quality multimedia applications.

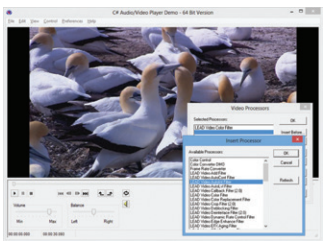
Publisher: LEADTOOLS | Category: Audio Visual | ★★★★★

LEADTOOLS Multimedia is specifically designed for the development of audio/video applications across a wide variety of industries, including defense, broadcast and security.



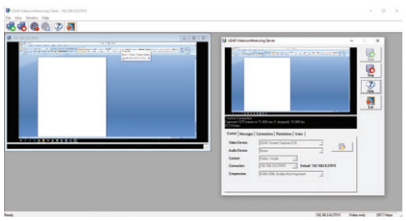
Conversion SDK Technology

Leverage DirectShow and Media Foundation to add audio and video conversion into your .NET (C# & VB) and C/C++ applications to fulfill requirements such as archival, Internet streaming, and mobile compatibility.



Multimedia Playback

LEADTOOLS offers a full range of flexible multimedia playback functionality all of which can be done for any multimedia source, including files, capture devices, TV tuners, discs, and network streams.



Video Conferencing

LEADTOOLS includes advanced video conferencing technology to develop DirectShow and Media Foundation applications that require video capture, playback, transmission, and efficient compression.

LEADTOOLS®
THE WORLD LEADER IN IMAGING SDKs

Prices from $ 975.10
www.componentsource.com/leadtools-multimedia

Nevron Vision for .NET

Top 100 Publisher

Corporate Partner

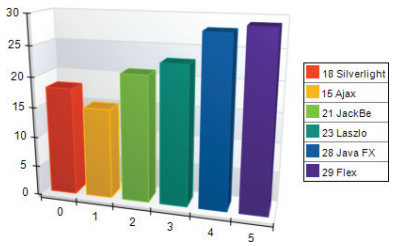
Data visualization components for Desktop and Web development.

Publisher: Nevron | Category: Presentation Layer

Nevron Vision for .NET offers everything you need to create advanced digital dashboards, reports, diagrams and MMI interfaces. It includes charts, gauges, diagrams, maps and user interface controls.

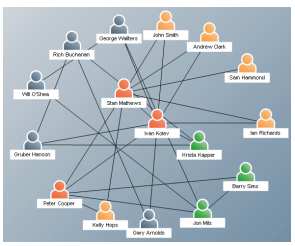
Nevron Gauge for .NET

Suitable for any application that needs to visualize KPIs or Scorecards. It features a complete set of Radial and Linear gauges, LED displays, State Indicators, Advanced Axes and visual effects.



Nevron Chart for .NET

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



Nevron Diagram for .NET

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

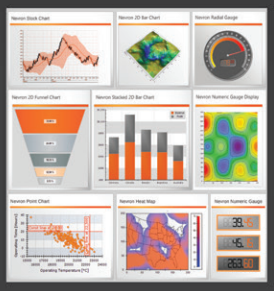
NEVRON

Prices from $ 901.55

www.componentsource.com/nevron-net-vision-ultimate

NEVRON

Nevron provides leading data visualization and UI controls for a diverse range of .NET centric platforms. Our controls and plugins integrate in desktop and web applications: WinForms, WPF, Silverlight, Xamarin, ASP.NET, SSRS, and Sharepoint.



Charts & Gauges

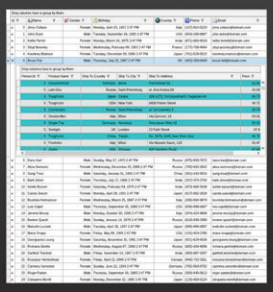
Our leading charting technology allows developers and IT professionals to create presentational, business, scientific, real-time monitoring, statistical, process control, BI, dashboards and other types of charts.
It includes features such as 2D/3D hardware acceleration, binding to a diverse range of data sources, advanced axes, numerous charting types and many more.

Diagrams

Industry leading diagramming components that can help you author and visualize complex flowcharts, org-charts, mind-maps, ERM, UML and other types of diagrams. Features include automatic layouts, interactive editing, routable connectors, groups, smart shapes, line jumps, Visio import, export to SVG and PDF and many more.

Text processing

Industry leading control for DOCX, RTF, PDF, HTML and EPUB document generation, editing and visualization. Our control allows you to create all types of text documents such as contracts, invoices, reports, and others. Suitable for advanced text processing, automated document generation, reporting, text document conversion, WYSIWYG text and HTML editing, proofreading and more.



Grids

High performance grid components for displaying, editing and analyzing tabular and hierarchical data. Features include binding to diverse data sources, grouping, filtering, sorting, master-details and advanced extensibility options.

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

Visual Studio Partner

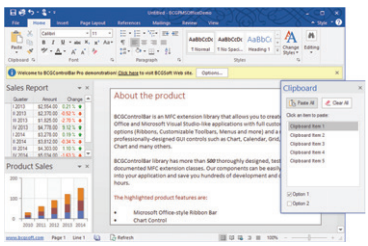
BCGControlBar Library Professional MFC

Top 50 Product | Top 25 Publisher | Professional Partner

Create Microsoft Office and Microsoft Visual Studio-like applications.

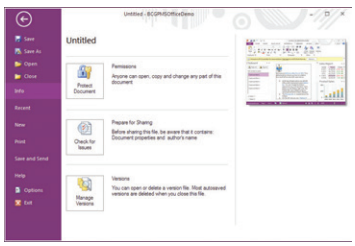
Publisher: BCGSoft | Category: UI & Interaction

Includes a rich set of GUI controls such as Chart, Calendar, Grid, Editor and Gantt. BCGControlBar Library has over 500 thoroughly designed, tested and fully documented MFC extension classes that will save you hundreds of development hours.



Ribbon Control

The Ribbon control replaces traditional toolbars and menus with tabbed groups (known as categories). Each tab is logically split into panels and each panel may contain various controls or command buttons.



Office Look & Feel

BCGControlBar Library provides a Microsoft Office look and feel. This is a skinned interface that can be applied to the various sets of controls such as ribbons, menus, toolbars, docking panes and more.

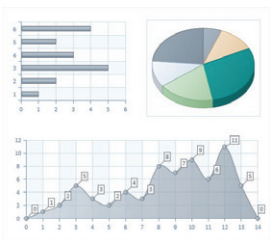


Chart Control

Create 2D and 3D charts with an unlimited number of series and unlimited data points. You can customize data markers and data labels, and data point values can be modified on the fly for real-time charting.

BCGSoft

Prices from $ 783.02

www.componentsource.com/bcgcontrolbar-library-professional

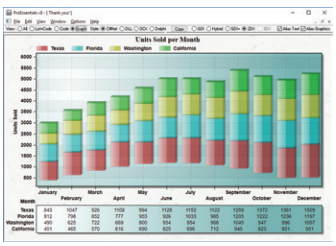
ProEssentials

Top 100 Product | Top 100 Publisher | Author

Add financial, scientific, engineering and business charts to your applications.

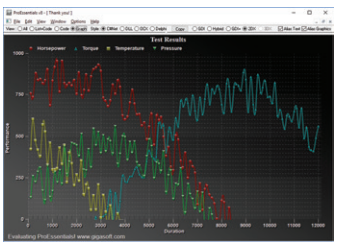
Publisher: GigaSoft | Category: Charts | ★★★★★

ProEssentials is ideal for real-time or continuous running chart implementations and includes a robust feature set, quality rendering, and simple implementation and distribution.



Graph Annotations

Annotations allow for mixing text and symbols in a tabular fashion within the chart or boundary of a chart. Table annotations can contain header columns and rows, hot spots, and real-time updating.



Scientific Graph

Pass large amounts of data into the graph and easily view the information in smaller/clearer increments via zooming or dialog adjustments. Zooming can be invoked by left-button dragging a zoom box.

Graph Types

Includes many differnet chart types: Line, Bar, Point/Scatter, Area, Points plus Best-Fit-Curve, Spline, High-Low Line, High-Low Bar, High-Low-Close, Open-High-Low-Close, Box Plot/ Candlestick, Bubble and Step.

GigaSoft

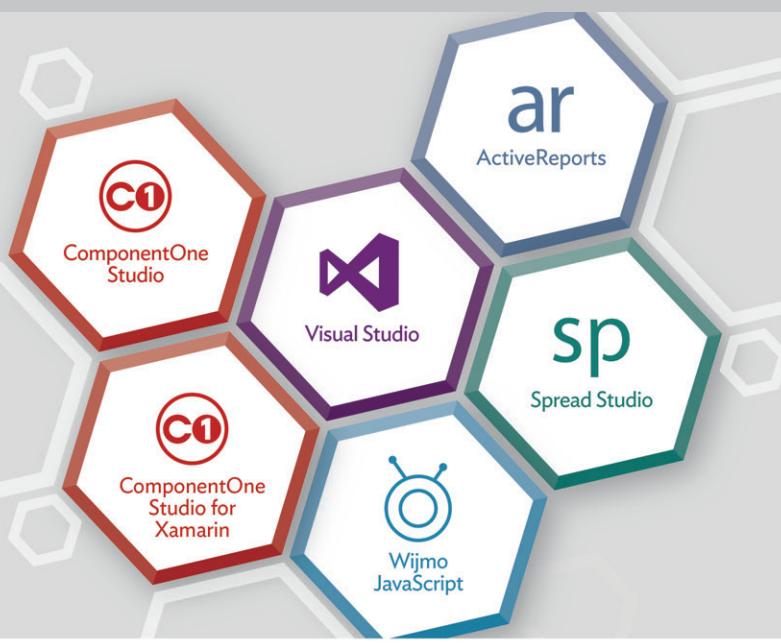
Prices from $ 1,763.02

www.componentsource.com/proessentials

GrapeCity

Empowering Developers To Deliver Apps Across All Platforms And Devices

GrapeCity's family of products provides developers with the most comprehensive collection of innovative, flexible, and easy-to-use UI, reporting, and spreadsheet controls for web, mobile, and desktop development.



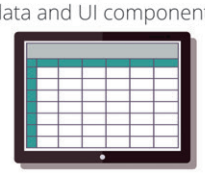
ComponentOne Studio
Flexible .NET data and UI controls

Wijmo
Extensible JavaScript/ HTML5 UI controls

ComponentOne Studio for Xamarin
Cross-platform native mobile UI controls

ActiveReports
Flexible and extensive reporting platform

Spread
Versatile spreadsheet data and UI components



Visual Studio Partner