



Your next great app starts here.

Download your FREE 30-day trial today at DevExpress.com/try



WinForms



ASP.NET



WPF



Silverlight



Windows 8 XAML



HTML JS



Reporting



DevExpress Mobile

msdn magazine

View

ViewModel

Model

Cross-Platform Presentation
with MVVM.....18

Writing a Cross-Platform Presentation Layer with MVVM Brent Edwards	18
Using the C++ REST SDK in Windows Store Apps Sridhar Poduri	32
Getting Started with Debugging Windows Store Apps Bruno Terkaly and Robert Evans	40
Upgrading Windows Phone 7.1 Apps to Windows Phone 8 Michael Crump	50
Adding FTP Support in Windows Phone 8 Uday Gupta	56

COLUMNS

DATA POINTS

Coding for Domain-Driven Design: Tips for Data-Focused Devs, Part 2

Julie Lerman, page 6

WINDOWS AZURE INSIDER

Hadoop and HDInsight: Big Data in Windows Azure

Bruno Terkaly and Ricardo Villalobos, page 12

TEST RUN

Multi-Swarm Optimization

James McCaffrey, page 68

THE WORKING PROGRAMMER

Exploring NSpec

Ted Neward, page 74

MODERN APPS

Understanding Your Language Choices for Developing Modern Apps

Rachel Appel, page 78

DIRECTX FACTOR

Direct2D Geometries and Their Manipulations

Charles Petzold, 82

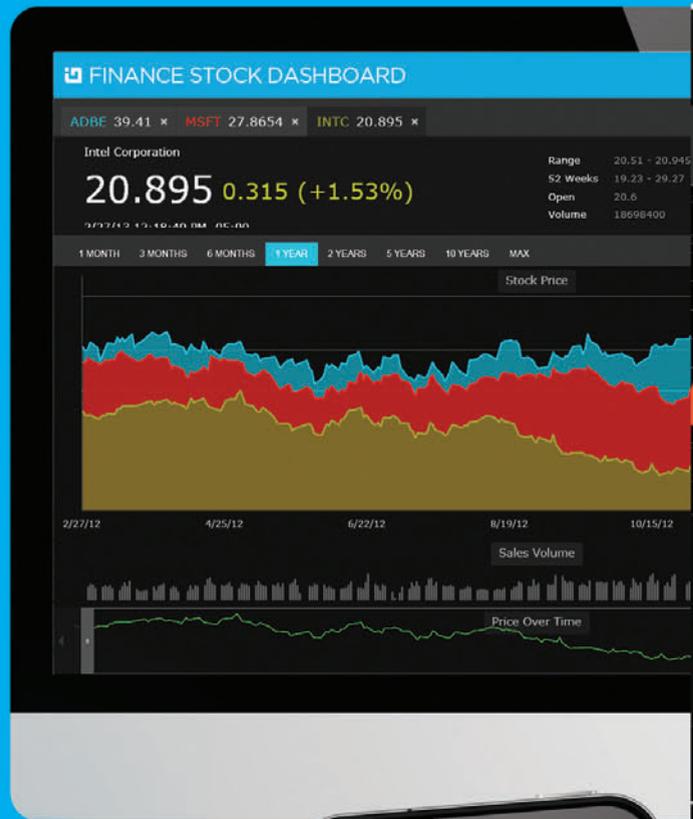
DON'T GET ME STARTED

Teenagers

David Platt, page 88

Desktop

Deliver high performance, scalable and stylable touch-enabled enterprise applications in the platform of your choice.



Native Mobile

Develop rich, device-specific user experience for iOS, Android, and Windows Phone, as well as mobile cross-platform apps with Mono-Touch.



Download Your Free Trial
infragistics.com/enterprise-READY



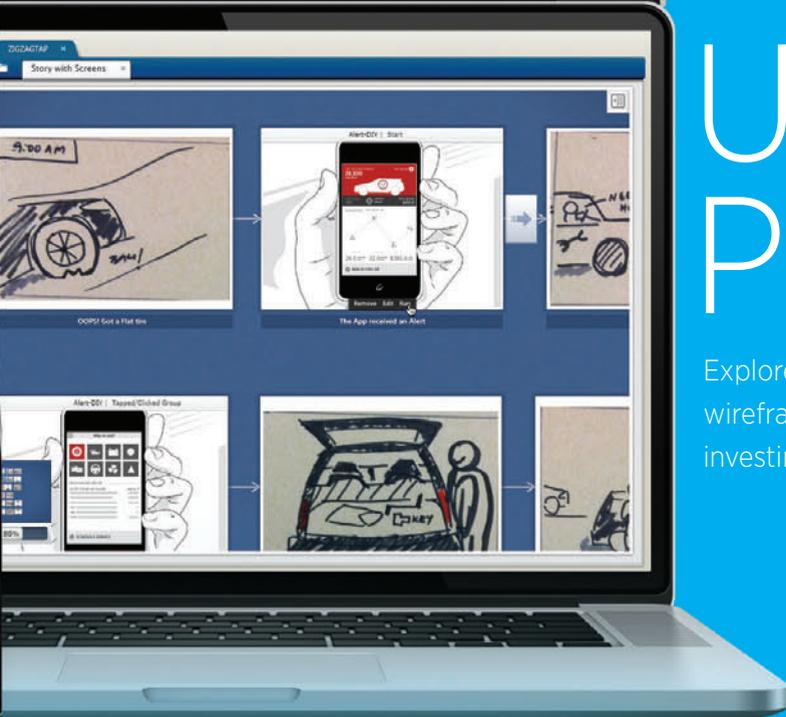
Cross-Device

Build standards-based, touch-enabled HTML5 & jQuery experiences for desktop, tablet, and mobile delivery, including multi-device targeting with frameworks such as PhoneGap and MVC.



UX Prototyping

Explore design ideas through rapid, user-centered wireframing, prototyping, and evaluation before investing a single line of code.



Infragistics Sales US 800 231 8588 • Europe +44 (0) 800 298 9055 • India +91 80 4151 8042 • APAC (+61) 3 9982 4545

Copyright 1996-2013 Infragistics, Inc. All rights reserved. Infragistics and NetAdvantage are registered trademarks of Infragistics, Inc. The Infragistics logo is a trademark of Infragistics, Inc.



dtSearch®

Instantly Search Terabytes of Text

- 25+ fielded and full-text search types
- dtSearch's **own document filters** support "Office," PDF, HTML, XML, ZIP, emails (with nested attachments), and many other file types
- Supports databases as well as static and dynamic websites
- **Highlights hits** in all of the above
- APIs for .NET, Java, C++, SQL, etc.
- 64-bit and 32-bit; Win and Linux

"lightning fast" Redmond Magazine

"covers all data sources" eWeek

"results in less than a second" InfoWorld

hundreds more reviews and developer case studies at www.dtsearch.com

dtSearch products:

- ◆ Desktop with Spider
- ◆ Web with Spider
- ◆ Network with Spider
- ◆ Engine for Win & .NET
- ◆ Publish (portable media)
- ◆ Engine for Linux
- ◆ Document filters also available for separate licensing

Ask about fully-functional evaluations

The Smart Choice for Text Retrieval® since 1991

www.dtSearch.com 1-800-IT-FINDS

msdn magazine

SEPTEMBER 2013 VOLUME 28 NUMBER 9

BJÖRN RETTIG Director

MOHAMMAD AL-SABT Editorial Director/mmeditor@microsoft.com

PATRICK O'NEILL Site Manager

MICHAEL DESMOND Editor in Chief/mmeditor@microsoft.com

DAVID RAMEL Technical Editor

SHARON TERDEMAN Features Editor

WENDY HERNANDEZ Group Managing Editor

KATRINA CARRASCO Associate Managing Editor

SCOTT SHULTZ Creative Director

JOSHUA GOULD Art Director

SENIOR CONTRIBUTING EDITOR Dr. James McCaffrey

CONTRIBUTING EDITORS Rachel Appel, Dino Esposito, Kenny Kerr, Julie Lerman, Ted Neward, Charles Petzold, David S. Platt, Bruno Terkaly, Ricardo Villalobos

Redmond Media Group

Henry Allain President, Redmond Media Group

Michele Imgrund Sr. Director of Marketing & Audience Engagement

Tracy Cook Director of Online Marketing

Irene Fincher Audience Development Manager

ADVERTISING SALES: 818-674-3416/dlbianca@1105media.com

Dan LaBianca Vice President, Group Publisher

Chris Kourtoglou Regional Sales Manager

Danna Vedder Regional Sales Manager/Microsoft Account Manager

Jenny Hernandez-Asandas Director, Print Production

Serena Barnes Production Coordinator/msdnadproduction@1105media.com

1105 MEDIA

Neal Vitale President & Chief Executive Officer

Richard Vitale Senior Vice President & Chief Financial Officer

Michael J. Valenti Executive Vice President

Christopher M. Coates Vice President, Finance & Administration

Erik A. Lindgren Vice President, Information Technology & Application Development

David F. Myers Vice President, Event Operations

Jeffrey S. Klein Chairman of the Board

MSDN Magazine (ISSN 1528-4859) is published monthly 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.

Printed in the U.S.A. Reproductions in whole or part prohibited except by written permission. Mail requests to "Permissions Editor," c/o MSDN Magazine, 4 Venture, Suite 150, Irvine, CA 92618.

Legal Disclaimer: The information in this magazine has not undergone any formal testing by 1105 Media, Inc. and is distributed without any warranty expressed or implied. Implementation or use of any information contained herein is the reader's sole responsibility. While the information has been reviewed for accuracy, there is no guarantee that the same or similar results may be achieved in all environments. Technical inaccuracies may result from printing errors and/or new developments in the industry.

Corporate Address: 1105 Media, Inc., 9201 Oakdale Ave., Ste 101, Chatsworth, CA 91311, www.1105media.com

Media Kits: Direct your Media Kit requests to Matt Morollo, VP Publishing, 508-532-1418 (phone), 508-875-6622 (fax), mmorollo@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

All customer service inquiries should be sent to MSDNmag@1105service.com or call 847-763-9560.



Printed in the USA

Introducing RaptorXML®

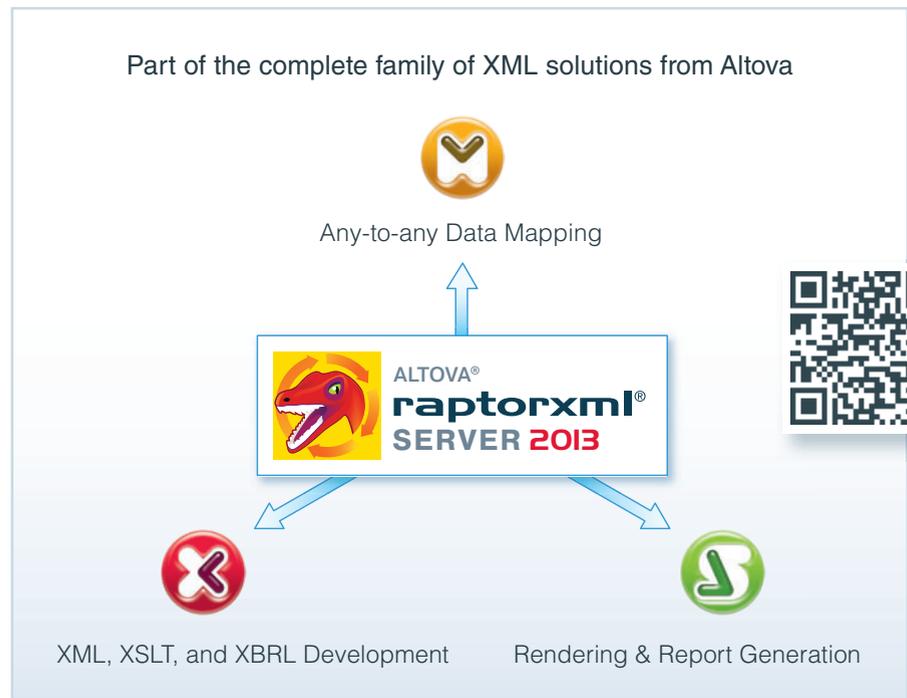
The new, hyper-fast XML and XBRL server
from the makers of XMLSpy®

- Ultra-high performance
- Ultra-low memory foot print
- Cross platform
- Optimized for parallel computing

Big Data trends and standards mandates are producing huge, ever increasing amounts of XML and XBRL data. Now, there is finally a modern, lightning-fast server to validate, process, transform, and query all of it: RaptorXML.

Standards Support:

XML 1.0, 1.1
XML Namespaces
XML Base
XInclude 1.0
XLink 1.0
XML Schema 1.0, 1.1
XPath 1.0, 2.0, 3.0
XSLT 1.0, 2.0, 3.0
XQuery 1.0, 3.0
Oasis Catalogs V1.1
XBRL 2.1
XBRL Dimensions
XBRL Formula 1.0
XBRL Functions
XBRL Definition Links



See the complete list of supported specifications and platforms, and download RaptorXML at

 www.altova.com/raptorxml



Standing Down

It was June 2012 when I decided to work standing up. A growing list of aches, pains and issues—the result of spending long hours every day in an office chair—motivated the move to a standing desk. And when I read that sitting all day just might kill you (bit.ly/jHRDqj), well, I couldn't get to my feet fast enough.

I came up with a low-cost arrangement that let me experiment with the new lifestyle. I had an extra 18-inch by 48-inch chrome wire shelf handy, which was just large enough to host my two LCD displays (one 27 inches, the other 24 inches) and a 12-inch laptop. I bought a set of 16-inch posts that raised the whole mess to a comfortable standing height when placed on my existing desk. A simple file box played host to my keyboard and mouse.

It took me a year to conclude what I've known all along: People aren't designed to stay in one position all day, whether that's sitting or standing.

As I wrote in my November 2012 column (msdn.microsoft.com/magazine/jj721586) and a June blog post (bit.ly/Nfuedz), the switch to a standing desk was awesome. I immediately felt better. Gone was the end-of-day numbness in my legs and pain in my right shoulder. My back, which was prone to intermittent aches and complaints, cleared up. My legs felt tired at the end of the day, but it was a *good*

tired. The act of standing not only did my body good, it helped make me more attentive, engaged and productive throughout the day.

And then things started to hurt. First, it was the feet—hardly a surprise to anyone who has worked in retail. I managed that by wearing shoes, standing on a padded mat, and placing a foot rest under the desk so I could switch off feet. Then my left knee started to complain. Pain and soreness, along with a pronounced bump, emerged around my left patellar tendon and kneecap.

Soon, a move intended to make me healthier was limiting my mobility. I avoided running and had to take care going up steps. Standing all day became exhausting and I found myself having to sit down or even lie down in the afternoons just to take pressure off my knees.

It took me a year to conclude what I've known all along: People aren't designed to stay in one position all day, whether that's sitting or standing. I've looked into a convertible setup of some sort, including variable-height desks and dual-monitor arm solutions (like those from Ergotron) that would let me raise and lower my displays. But so far I haven't seen anything affordable enough (less than, say, \$500) that will do the job.

Not that I'm going to stay sitting for long. I can already feel the old twinges returning in my shoulder and back, and I worry about the accumulating health impacts of sitting all day. So I plan to restore my jury-rigged standing desk setup, and augment it with a drafting chair that will let me take occasional sit breaks while working at the desk.

Have you moved to a standing desk? And if so, how did you manage the transition? E-mail me at mmeditor@microsoft.com.

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

© 2013 Microsoft Corporation. All rights reserved.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, you are not permitted to reproduce, store, or introduce into a retrieval system *MSDN Magazine* or any part of *MSDN Magazine*. If you have purchased or have otherwise properly acquired a copy of *MSDN Magazine* in paper format, you are permitted to physically transfer this paper copy in unmodified form. Otherwise, you are not permitted to transmit copies of *MSDN Magazine* (or any part of *MSDN Magazine*) in any form or by any means without the express written permission of Microsoft Corporation.

A listing of Microsoft Corporation trademarks can be found at microsoft.com/library/toolbar/3.0/trademarks/en-us.mspx. Other trademarks or trade names mentioned herein are the property of their respective owners.

MSDN Magazine is published by 1105 Media, Inc. 1105 Media, Inc. is an independent company not affiliated with Microsoft Corporation. Microsoft Corporation is solely responsible for the editorial contents of this magazine. The recommendations and technical guidelines in *MSDN Magazine* are based on specific environments and configurations. These recommendations or guidelines may not apply to dissimilar configurations. Microsoft Corporation does not make any representation or warranty, express or implied, with respect to any code or other information herein and disclaims any liability whatsoever for any use of such code or other information. *MSDN Magazine*, MSDN, and Microsoft logos are used by 1105 Media, Inc. under license from owner.

the #1 selling scrum software

The screenshot displays the OnTime web application interface. The main area is a Scrum board with columns: New Request, Approved, In Progress, Completed, and Rejected. Each card represents a task with details like assignee, priority, and release date. A sidebar on the left provides navigation for Projects, Releases, and Team Members. An inset window titled 'OnTime Visual Studio Extension' shows a task list with columns for ID, Name, Assignee, Priority, and Status.



OnTime Scrum agile project management software

Want to ship your software 24% faster? Think about it: that's **3 free months every year**, and it's the average time saved by users of OnTime Scrum.

How are they doing it? Because OnTime Scrum is fast, extremely customizable, and easy to use. Features like **Card View** make managing your user stories highly visible and effortless. And with our **Visual Studio integration**, your dev team can manage their OnTime items without leaving their development environment.

So is it any wonder that OnTime Scrum is the **#1 selling scrum software**?

just **\$7** per user per month

Get 10% off your OnTime subscription with this link:
OnTimeNow.com/MSDN

Plus: learn about our solutions for bug tracking, help desk & customer management, and project wikis





Coding for Domain-Driven Design: Tips for Data-Focused Devs, Part 2

This month's column will continue celebrating the 10th anniversary of Eric Evans' book, "Domain-Driven Design: Tackling Complexity in the Heart of Software" (Addison-Wesley Professional, 2003). I'll share more tips for data-first developers who are interested in benefiting from some of the coding patterns of Domain-Driven Design (DDD). Last month's column highlighted:

- Setting persistence concerns aside when modeling the domain
- Exposing methods to drive your entities and aggregates, but not your property setters
- Recognizing that some subsystems are perfect for simple create, read, update and delete (CRUD) interaction and don't require domain modeling
- The benefit of not attempting to share data or types across bounded contexts

In this column you'll learn what's meant by the terms "anemic" and "rich" domain models, as well as the concept of value objects. Value objects seem to be a topic that falls into one of two camps for developers. It's either so obvious that some readers can't imagine why it warrants any discussion at all, or so baffling that others have never been able to wrap their heads around it. I'll also encourage you to consider using value objects in place of related objects in certain scenarios.

That Condescending Term: Anemic Domain Models

There's a pair of terms you'll often hear with respect to how classes are defined in DDD—anemic domain model and rich domain model. In DDD, domain model refers to a class. A rich domain model is one that aligns with the DDD approach—a class (or type) that's defined with behaviors, not just with getters and setters. In contrast, an anemic domain model consists simply of getters and setters (and maybe a few simple methods), which is fine for many scenarios, but gains no benefit from DDD.

When I started using Entity Framework (EF) Code First, the classes I intended Code First to work with were probably 99 percent anemic. **Figure 1** is a perfect example, showing a Customer type and the Person type from which it inherits. I'd often include a handful of methods, but the basic type was merely a schema with getters and setters.

I finally looked at these classes and realized I was doing little more than defining a database table in my class. It's not necessarily a bad thing, depending on what you plan to do with the type.

Compare that to the richer Customer type listed in **Figure 2**, which is similar to the Customer type I explored in my previous column

(msdn.microsoft.com/magazine/dn342868). It exposes methods that control access to properties and other types that are part of the aggregate. I've tweaked the Customer type since you saw it in the earlier column to better express some of the topics I'm discussing this month.

In this richer model, rather than simply exposing properties to be read and written to, the public surface of Customer is made up of explicit methods. Review the Private Setters and Public Methods section of last month's column for more details. My point here is to help you better comprehend the difference between what DDD refers to as an anemic versus a rich domain model.

Value Objects Can Be Confusing

Though they seem simple, DDD value objects are a serious point of confusion for many, including me. I've read and heard so many different ways of describing value objects from a variety of perspectives. Luckily, each of the different explanations, rather than conflicting with one another, helped me build a deeper understanding of value objects.

At its core, a value object is a class that doesn't have an identity key.

Entity Framework has a concept of "complex types" that comes fairly close. Complex types don't have an identity key—they're a way of encapsulating a set of properties. EF understands how to handle these objects when it comes to data persistence. In the database, they get stored as fields of the table to which the entity is mapped.

For example, instead of having a FirstName property and a LastName property in your Person class, you could have a FullName property:

```
public FullName FullName { get; set; }
```

Figure 1 Typical Anemic Domain Model Classes Look Like Database Tables

```
public class Customer : Person
{
    public Customer()
    {
        Orders = new List<Order>();
    }
    public ICollection<Order> Orders { get; set; }
    public string SalesPersonId { get; set; }
    public ShippingAddress ShippingAddress { get; set; }
}

public abstract class Person
{
    public int Id { get; set; }
    public string Title { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public string CompanyName { get; set; }
    public string EmailAddress { get; set; }
    public string Phone { get; set; }
}
```

Code download available at archive.msdn.microsoft.com/mag201309DataPoints.

NEW OPPORTUNITIES WITH NEW DOMAINS

Choose from **over 500 new top-level domains!** Create a short, memorable web address that perfectly fits your business or brand, such as **your-name.blog**, **auto.shop** or **events.nyc**. You can also make your website easier to find by getting new extensions for your existing domain.

With 1&1, it is easy to connect a registered domain to any website, no matter which provider is hosting your website.

Find out more at 1and1.com

NEW!
PRE-RESERVE
FREE
WITH NO OBLIGATION!*



DOMAINS | E-MAIL | WEB HOSTING | eCOMMERCE | SERVERS

1and1.com

* Pre-reserving a domain name does not guarantee that you will be able to register that domain. Other terms and conditions may apply. Visit www.1and1.com for full promotional offer details. Program and pricing specifications and availability are subject to change without notice. 1&1 and the 1&1 logo are trademarks of 1&1 Internet, all other trademarks are the property of their respective owners. ©2013 1&1 Internet. All rights reserved.

FullName can be a class that encapsulates the FirstName and LastName properties, with a constructor that forces you to provide both of those properties.

But a value object is more than a complex type. In DDD, there are three defining characteristics of a value object:

1. It has no identity key (this aligns with a complex type)
2. It's immutable
3. When checking its equality to other instances of the same type, all of its values are compared

Immutability is interesting. Without an identity key, a type's immutability defines its identity. You can always identify an instance by the combination of its properties. Because it's immutable, none of the type's properties can change, so it's safe to use the type's values to identify a particular instance. (You've already seen how DDD entities protect themselves from random modifications by making their setters private, but you could have a method that allows you to affect

Figure 2 A Customer Type That's a Rich Domain Model, Not Simply Properties

```
public class Customer : Contact
{
    public Customer(string firstName, string lastName, string email)
    {
        FullName = new FullName(firstName, lastName);
        EmailAddress = email;
        Status = CustomerStatus.Silver;
    }

    internal Customer()
    {
    }

    public void UseBillingAddressForShippingAddress()
    {
        ShippingAddress = new Address(
            BillingAddress.Street1, BillingAddress.Street2,
            BillingAddress.City, BillingAddress.Region,
            BillingAddress.Country, BillingAddress.PostalCode);
    }

    public void CreateNewShippingAddress(string street1, string street2,
        string city, string region, string country, string postalCode)
    {
        BillingAddress = new Address(
            street1,street2,
            city,region,
            country,postalCode)
    }

    public void CreateBillingInformation(string street1,string street2,
        string city,string region,string country, string postalCode,
        string creditcardNumber, string bankName)
    {
        BillingAddress = new Address
            (street1,street2, city,region,country,postalCode );
        CreditCard = new CustomerCreditCard (bankName, creditcardNumber );
    }

    public void SetCustomerContactDetails
        (string email, string phone, string companyName)
    {
        EmailAddress = email;
        Phone = phone;
        CompanyName = companyName;
    }

    public string SalesPersonId { get; private set; }
    public CustomerStatus Status { get; private set; }
    public Address ShippingAddress { get; private set; }
    public Address BillingAddress { get; private set; }
    public CustomerCreditCard CreditCard { get; private set; }
}
```

those properties.) Not only does a value object hide the setters, but it prevents you from modifying any of the properties. If you modified a property, that would mean the value of the object has changed. Because the entire object represents its value, you don't interact with its properties individually. If you need the object to have different values, you create a new instance of the object to hold a new set of values. In the end, there's no such thing as changing a property of a value object—which effectively makes even formulating a sentence that includes the phrase “if you need to change a value of a property” an oxymoron. A useful parallel is to consider a string, which is another immutable type (at least in all languages with which I'm familiar). When working with a string instance, you don't replace individual characters of the string. You simply create a new string.

My FullName value object is shown in **Figure 3**. It has no identity key property. You can see that the only way to instantiate it is by providing both property values, and there's no way to modify either of those properties. So it fulfills the immutability requirement. The final requirement, that it provides a way to compare equality to another instance of that type, is hidden in the custom ValueObject class (that I've borrowed from Jimmy Bogard at bit.ly/13SWd9h) from which it inherits, because that's a bunch of convoluted code. Although this ValueObject doesn't account for collection properties, it meets my needs because I don't have any collections within this value object.

Keeping in mind that you may want a modified copy—for example, similar to this FullName but with a different FirstName—Vaughn Vernon, author of “Implementing Domain-Driven Design” (Addison-Wesley Professional, 2013), suggests FullName could include methods to create new instances from the existing instance:

```
public FullName WithChangedFirstName(string firstName)
{
    return new FullName(firstName, this.LastName);
}
public FullName WithChangedLastName(string lastName)
{
    return new FullName(this.FirstName, lastName);
}
```

When I add in my persistence layer (Entity Framework), it will see this as an EF ComplexType property of any class in which it's used. When I use FullName as a property of my Person type, EF will store the properties of FullName in the database table where the Person type is stored. By default, the properties will be named FullName_FirstName and FullName_LastName in the database.

Figure 3 The FullName Value Object

```
public class FullName:ValueObject<FullName>
{
    public FullName(string firstName, string lastName)
    {
        FirstName = firstName;
        LastName = lastName;
    }

    public FullName(FullName fullName)
        : this(fullName.FirstName, fullName.LastName)
    {
    }

    internal FullName() { }

    public string FirstName { get; private set; }
    public string LastName { get; private set; }

    // More methods, properties for display formatting
}
```

Empower Your Customers



Create & Edit PDFs in .Net - ActiveX - WinRT

- 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 WinRT Component enables publishing C#, C++CX or Javascript apps to Windows Store
- New 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 certified PDF Converter printer driver
- Export PDF documents into other formats such as JPeg, PNG, XAML or HTML5

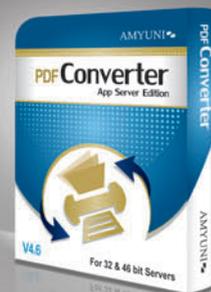


Advanced HTML to PDF & XAML

- Direct conversion of HTML files into PDF and XAML without the use of a web browser or a printer driver
- Easy Integration and deployment within developer's applications
- WebkitPDF is based on the Webkit Open Source library and Amyuni PDF Creator



High Performance PDF Printer For Desktops and Servers



- Our high-performance printer driver optimized for Web, Application and Print Servers. Print to PDF in a fraction of the time needed with other tools. WHQL tested for Windows 32 and 64-bit including Windows Server 2012 and Windows 8
- Standard PDF features included with a number of unique features. Interface with any .Net or ActiveX programming language
- Easy licensing and deployment to fit system administrator's requirements



AMYUNI

All development tools available at

www.amyuni.com

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

FullName is pretty straightforward. Even if you were doing database-focused design, you might not want to store FirstName and LastName in a separate table.

Value Objects or Related Objects?

But now consider another scenario—imagine a Customer with a ShippingAddress and a BillingAddress:

```
public Address ShippingAddress { get; private set; }
public Address BillingAddress { get; private set; }
```

By (data-driven brain) default, I create Address as an Entity, including an AddressId property. And, again, because I “think in data,” I presume that Address will be stored as a separate table in the database. OK, I’m working hard to train myself to stop considering the database while modeling my domain, so that’s of no consequence. However, at some point I’ll add in my data layer using EF and EF will also make this assumption. But EF won’t be able to work out the mappings correctly. It will presume a 0..1:* relationship between Address and Customer; in other words, that an address might have any number of related customers and a Customer will have either no addresses or one address.

My DBA might not like this result and, what’s more important, I won’t be writing my application code based on the same assumption that Entity Framework is making—that there are many Customers to zero or one Address. Because of this, EF might affect my data persistence or retrieval in some unexpected ways. So from the perspective of someone who’s been working with EF quite a lot, my first approach is to fix the EF mappings using the Fluent API. But asking EF to fix this problem would force me to add navigation properties from Address pointing back to Customer, which I don’t want in my model. As you can see, solving this problem with EF mappings would lead me down a rabbit hole.

When I step back and focus on the domain, not on EF or the database, it makes more sense to simply make Address a value object instead of an entity. Three steps are required:

1. I need to remove the key property from the Address type (possibly called Id or AddressId).
2. I need to be sure that Address is immutable. Its constructor already lets me populate all of the fields. I have to remove any methods that would allow any properties to change.

Figure 4 Address as a Value Object

```
public class Address:ValueObject<Address>
{
    public Address(string street1, string street2, string city, string region,
        string country, string postalCode) {
        Street1 = street1;
        Street2 = street2;
        City = city;
        Region = region;
        Country = country;
        PostalCode = postalCode; }

    internal Address() { }

    public string Street1 { get; private set; }
    public string Street2 { get; private set; }
    public string City { get; private set; }
    public string Region { get; private set; }
    public string Country { get; private set; }
    public string PostalCode { get; private set; }

    // ...
}
```

3. I need to ensure that Address can be checked for equality based on the values of its properties and fields. I can do that by inheriting again from that nice ValueObject class I borrowed from Jimmy Bogard (see Figure 4).

How I use this Address class from my Customer class won’t change, except that if I need to modify the address, I’ll have to create a new instance.

But now I no longer need to worry about managing that relationship. And it also provides another litmus test for value objects, in that a Customer really is defined by its shipping and billing information because if I sell anything to that customer, I’ll most likely need to ship it and the Accounts Payable department requires that I have its billing address.

This is not to say that every 1:1 or 1:0..1 relationship can be replaced with value objects, but it’s a nice solution here and my job became a lot simpler when I stopped having to solve one puzzle after another that popped up as I tried to force Entity Framework to maintain this relationship for me.

Like the FullName example, changing Address to be a value object means Entity Framework will see Address as a ComplexType and will store all of that data within the Customer tables. Years of being focused on database normalization made me automatically think this was a bad thing, but my database can handle it easily and it works for my particular domain.

I can think of many arguments against using this technique, but they all begin with “what if.” Any of those that don’t pertain to my domain aren’t valid arguments. We waste a lot of time coding for just-in-case scenarios that never come to pass. I’m trying to be more considerate about adding those preemptive Band-Aids into my solutions.

Not Done Quite Yet

I’m really getting through the list of DDD concepts that have haunted this data geek. As I grok more of these concepts, I become inspired to learn even more. Once I shift my thinking a bit, these patterns make a lot of sense for me. They don’t reduce my interest in data persistence one bit, but separating the domain from the data persistence and infrastructure feels right to me after having them tangled up together for so many years. Still, it’s important to keep in mind that there are so many software activities that don’t need DDD. DDD can help sort out complex problems, but quite often it’s overkill for simple ones.

In the next column I’ll discuss a few other DDD technical strategies that also initially seemed to conflict with my data-first thinking, such as letting go of bidirectional relationships, considering invariants and dealing with any perceived need to trigger data access from your aggregates. ■

JULIE LERMAN is a Microsoft MVP, .NET mentor and consultant who lives in the hills of Vermont. You can find her presenting on data access and other Microsoft .NET topics at user groups and conferences around the world. She blogs at thedatafarm.com/blog and is the author of “Programming Entity Framework” (2010) as well as a Code First edition (2011) and a DbContext edition (2012), all from O’Reilly Media. Follow her on Twitter at twitter.com/julielerman and see her Pluralsight courses at julieme/PS-Videos.

THANKS to the following technical expert for reviewing this article:
Stephen Bohlen (Microsoft)

Telerik DevCraft

The all-in-one toolset for professional developers targeting Microsoft platforms.



- Create web, mobile and desktop applications that impress
- Cover any .NET platform and any device
- Code faster and smarter without cutting corners

Get your 30-day free trial today:
www.telerik.com/all-in-one

 **telerik**





Hadoop and HDInsight: Big Data in Windows Azure

Let's begin with a bold assertion: "If you, your startup, or the enterprise you work for aren't saving massive quantities of data to disk for current and future analysis, you are compromising your effectiveness as a technical leader." Isn't it foolish to base important business decisions on gut instinct alone, rather than real quantitative data?

There are many reasons why big data is so pervasive. First, it's amazingly cheap to collect and store data in any form, structured or unstructured, especially with the help of products such as Windows Azure Storage services. Second, it's economical to leverage the cloud to provide the needed compute power—running on commodity hardware—to analyze this data. Finally, big data done well provides a major competitive advantage to businesses because it's possible to extract undiscovered information from vast quantities of unstructured data. The purpose of this month's article is to show how you can leverage the Windows Azure platform—in particular the Windows Azure HDInsight Service—to solve big data challenges.

Barely a day goes by without some hyped-up story in the IT press—and sometimes even in the mainstream media—about big data. Big data refers simply to data sets so large and complex that they're difficult to process using traditional techniques, such as data cubes, de-normalized relational tables and batch-based extract, transform and load (ETL) engines, to name a few. Advocates talk about extracting business and scientific intelligence from petabytes of unstructured data that might originate from a variety of sources: sensors, Web logs, mobile devices and the Internet of Things, or IoT (technologies based on radio-frequency identification [RFID], such as near-field communication, barcodes, Quick Response [QR] codes and digital watermarking). IoT changes the definition of big—we're now talking about exabytes of data a day!

Does big data live up to all the hype? Microsoft definitely believes it does and has bet big on big data. First, big data leads to better marketing strategies, replacing gut-based decision making with analytics based on real consumer behavior. Second, business leaders can improve strategic decisions, such as adding a new feature to an application or Web site, because they can study the telemetry and usage data of applications running on a multitude of devices. Third, it helps financial services detect fraud and assess risk. Finally, though you may not realize it, it's big data technologies that are typically used to build recommendation engines (think Netflix). Recommendations are often offered as a service on the Web or within big companies to expedite business decisions. The really smart businesses are collecting data today without even knowing what type of questions they're going to ask of the data tomorrow.

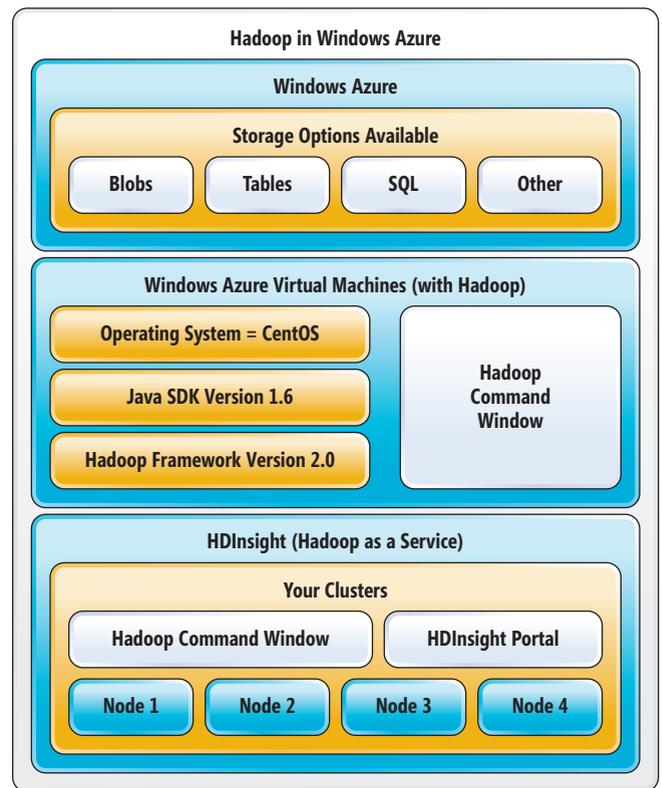


Figure 1 Hadoop Ecosystem in Windows Azure

Big data really means data analytics, which has been around for a long time. While there have always been huge data stores being mined for intelligence, what makes today's world different is the sheer variety of mostly unstructured data. Fortunately, products like Windows Azure bring great economics, allowing just about anyone to scale up his compute power and apply it to vast quantities of storage, all in the same datacenter. Data scientists describe the new data phenomenon as the three Vs—velocity, volume and variety. Never has data been created with such speed, size and the lack of a defined structure.

The world of big data contains a large and vibrant ecosystem, but one open source project reigns above them all, and that's Hadoop. Hadoop is the de facto standard for distributed data crunching. You'll find a great introduction to Hadoop at bit.ly/PPGvDP. "Hadoop provides a MapReduce framework for writing applications that process large amounts of structured and semi-structured data

Figure 2 Crime Data Summarized by Type

Grand Theft from Locked Auto	2617
Malicious Mischief	1623
Drivers License	1230
Aided Case	1195
Lost Property	1083

in parallel across large clusters of machines in a very reliable and fault-tolerant manner.” In addition, as you learn more about this space, you’ll likely come to agree with the perspective of Matt Winkler (principal PM on HDInsight) that Hadoop is “an ecosystem of related projects on top of the core distributed storage and MapReduce framework.” Paco Nathan, director of data science at Concurrent and a committer on the Cascading open source project (cascading.org), says further, “the abstraction layers enable people to leverage Hadoop at scale without knowing the underpinnings.”

The MapReduce Model

MapReduce is the programming model used to process huge data sets; essentially it’s the “assembly language” for Hadoop, so understanding what it does is crucial to understanding Hadoop. MapReduce algorithms are written in Java and split the input data set into independent chunks that are processed by the map tasks in a completely parallel manner. The framework sorts the output of the maps, which are then input to the reduce tasks. Typically, both the input and the output of the job are stored in a file system. The framework takes care of scheduling tasks, monitoring them and re-executing failed tasks.

Ultimately, most developers won’t author low-level Java code for MapReduce. Instead, they’ll use advanced tooling that abstracts the complexities of MapReduce, such as Hive or Pig. To gain an appreciation of this abstraction, we’ll take a look at low-level Java MapReduce and at how the high-level Hive query engine, which HDInsight supports, makes the job much easier.

Why HDInsight?

HDInsight is an Apache Hadoop implementation that runs in globally distributed Microsoft datacenters. It’s a service that allows you to easily build a Hadoop cluster in minutes when you need it, and tear it down after you run your MapReduce jobs. As Windows Azure Insiders, we believe there are a couple key value propositions of HDInsight. The first is that it’s 100 percent Apache-based, not a special Microsoft version, meaning that as Hadoop evolves, Microsoft will embrace the newer versions. Moreover, Microsoft is a major contributor to the Hadoop/Apache project and has provided a great deal of its query optimization know-how to the query tooling, Hive.

The second aspect of HDInsight that’s compelling is that it works seamlessly with Windows Azure Blobs, mechanisms for storing large amounts of unstructured data that can be accessed from anywhere in the world via HTTP or HTTPS. HDInsight also makes it possible to persist the meta-data of table definitions in SQL Server so that when the cluster is shut down, you don’t have to re-create your data models from scratch.

Figure 1 depicts the breadth and depth of Hadoop support in the Windows Azure platform.

On top is the Windows Azure Storage system, which provides secure and reliable storage and includes built-in geo-replication for redundancy of your data across regions. Windows Azure Storage includes a variety of powerful and flexible storage mechanisms, such as Tables (a NoSQL, keyvalue store), SQL database, Blobs and more. It supports a REST-ful API that allows any client to perform create, read, update, delete (CRUD) operations on text or binary data, such as video, audio and images. This means that any HTTP-capable client can interact with the storage system. Hadoop directly interacts with Blobs, but that doesn’t limit your ability to leverage other storage mechanisms within your own code.

The second key area is Windows Azure support for virtual machines (VMs) running Linux. Hadoop runs on top of Linux and leverages Java, which makes it possible to set up your own single-node or multi-node Hadoop cluster. This can be a tremendous money saver and productivity booster, because a single VM in Windows Azure is very economical. You can actually build your own multi-node cluster by hand, but it isn’t trivial and isn’t needed when you’re just trying to validate some basic algorithms.

Setting up your own Hadoop cluster makes it easy to start learning and developing Hadoop applications. Moreover, performing the setup yourself provides valuable insight into the inner workings of a Hadoop job. If you want to know how to do it, see the blog post, “How to Install Hadoop on a Linux-Based Windows Azure Virtual Machine,” at bit.ly/1am85mU.

HDInsight is an Apache Hadoop implementation that runs in globally distributed Microsoft datacenters.

Of course, once you need a larger cluster, you’ll want to take advantage of HDInsight, which is available today in Preview mode. To begin, go into the Windows Azure portal (bit.ly/12jt5KW) and sign in. Next, select Data Services | HDInsight | Quick Create. You’ll be asked for a cluster name, the number of compute nodes (currently four to 32 nodes) and the storage account to which to bind. The location of your storage account determines the location of your cluster. Finally, click CREATE HDINSIGHT CLUSTER. It will take 10 to 15 minutes to provision your cluster. The time it takes to provision is not related to the size of the cluster.

Note that you can also create and manage an HDInsight cluster programmatically using Windows PowerShell, as well as through cross-platform tooling on Linux- and Mac-based systems. Much of the functionality in the command-line interface (CLI) is also available in an easy-to-use management portal, which allows you to manage the cluster, including the execution and management of jobs on the cluster. You can download Windows Azure

Figure 3 The MapReduce Java Code That Summarizes the Crime Data

```
// CrimeCount.java
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;

// This code is based on the standard word count examples
// you can find almost everywhere.
// We modify the map function to be able to aggregate based on
// crime type.
// The reduce function as well as main is unchanged,
// except for the name of the job.

public class CrimeCount {

    public static class Map extends MapReduceBase implements
        Mapper<LongWritable, Text, Text, IntWritable> {
        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();

        private String mytoken = null;
        public void map(LongWritable key, Text value,
            OutputCollector<Text, IntWritable> output,
            Reporter reporter) throws IOException {
            // Read one line from the input file.
            String line = value.toString();
            // Parse the line into separate columns.
            String[] myarray = line.split(",");
            // Verify that there are at least three columns.
            if(myarray.length >= 2){
                // Grab the third column and increment that
                // crime (i.e. LOST PROPERTY found, so add 1).
                mytoken = myarray[2];
                word.set(mytoken);
                // Add the key/value pair to the output.
                output.collect(word, one);
            }
        }
    }

    // A fairly generic implementation of reduce.
    public static class Reduce extends MapReduceBase implements Reducer<Text,
        IntWritable, Text, IntWritable> {

        public void reduce(Text key, Iterator<IntWritable> values,
            OutputCollector<Text,
            IntWritable> output,
            Reporter reporter) throws IOException {

            // Loop through an aggregate key/value pairs.
            int sum = 0;
            while (values.hasNext()) {
                sum += values.next().get();
            }
            output.collect(key, new IntWritable(sum));
        }
    }

    // Kick off the MapReduce job, specifying the map and reduce
    // construct, as well as input and output parameters.
    public static void main(String[] args) throws Exception {
        JobConf conf = new JobConf(CrimeCount.class);
        conf.setJobName("crimecount");

        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(IntWritable.class);

        conf.setMapperClass(Map.class);
        conf.setCombinerClass(Reduce.class);
        conf.setReducerClass(Reduce.class);

        conf.setInputFormat(TextInputFormat.class);
        conf.setOutputFormat(TextOutputFormat.class);

        FileInputFormat.setInputPaths(conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(conf, new Path(args[1]));

        JobClient.runJob(conf);
    }
}
```

PowerShell as well as the CLIs for Mac and Linux at bit.ly/ZueX9Z. Then set up your VM running CentOS (a version of Linux), along with the Java SDK and Hadoop.

Exploring Hadoop

To experiment with Hadoop and learn about its power, we decided to leverage publicly available data from data.sfgov.org. Specifically, we downloaded a file containing San Francisco crime data for the previous three months and used it as is. The file includes more

than 33,000 records (relatively small by big data standards) derived from the SFPD Crime Incident Reporting system. Our goal was to perform some simple analytics, such as calculating the number and type of crime incidents. **Figure 2** shows part of the output from the Hadoop job that summarized the crime data.

The code in **Figure 3** summarizes the three months of crimes. The input file contained more than 30,000 rows of data, while the output contained only 1,000 records. The top five of those 1,000 records are shown in **Figure 2**.

Once you've saved the code in **Figure 3** as `CrimeCount.java`, you need to compile, package and submit the Hadoop job. **Figure 4** contains instructions for copying the input crime data file into the Hadoop Distributed File System (HDFS); compiling `CrimeCount.java`; creating the `crimecount.jar` file; running the Hadoop job (using `crimecount.jar`); and viewing the results—that is, the output data. To download the entire source code, go to sdr.ms/16kKJKh and right-click the `CrimeCount` folder.

Now you have an idea of the pieces that make up a minimal Hadoop environment, as well as what MapReduce Java code looks like and how it ends up being submitted as a Hadoop job at the command line. Chances are, at some point you'll want spin up a cluster to run some big jobs, then shut it down using higher-level tooling like Hive or Pig, and this is what HDInsight is all about because it makes it easy, with built-in support for Pig and Hive.

Once your cluster is created, you can work at the Hadoop command prompt or you can use the portal to issue Hive and Pig queries. The

Figure 4 Compiling, Packaging and Running the Hadoop Job

```
# Make a folder for the input file.
hadoop fs -mkdir /tmp/hadoopjob/crimecount/input
# Copy the data file into the folder.
hadoop fs -put SFPD_Incidents.csv /tmp/hadoopjob/crimecount/input
# Create a folder for the Java output classes.
mkdir crimecount_classes
# Compile the Java source code.
javac -classpath /usr/lib/hadoop/hadoop-common-2.0.0-cdh4.3.0.jar:/usr/lib/hadoop-0.20-mapreduce/hadoop-core-2.0.0-mr1-cdh4.3.0.jar -d crimecount_classes CrimeCount.java
# Create a jar file from the compiled Java code.
jar -cvf crimecount.jar -C crimecount_classes/ .
# Submit the jar file as a Hadoop job, passing in class path as well as
# the input folder and output folder.
# *NOTE* HDInsight users can use "asv:///SFPD_Incidents.csv," instead of
# "/tmp/hadoopjob/crimecount/input" if they uploaded the input file
# (SFPD_Incidents.csv) to Windows Azure Storage.
hadoop jar crimecount.jar org.myorg.CrimeCount /tmp/hadoopjob/crimecount/
input /tmp/hadoopjob/crimecount/output
# Display the output (the results) from the output folder.
hadoop fs -cat /tmp/hadoopjob/crimecount/output/part-00000
```

WPF lives!



➔ **XCEED Business Suite for WPF**

The essential set of WPF controls for all your line-of-business solutions. Includes the industry-leading **Xceed DataGrid for WPF**.
A total of 85 tools!

Figure 5 Hive Query Code to Perform the MapReduce

```
# Hive does a remarkable job of representing native Hadoop data stores
# as relational tables so you can issue SQL-like statements.

# Create a pseudo-table for data loading and querying
CREATE TABLE sfpdcrime(
  IncidntNum string,
  Category string,
  Descript string,
  DayOfWeek string,
  CrimeDate string,
  CrimeTime string,
  PdDistrict string,
  Resolution string,
  Address string,
  X string,
  Y string,
  CrimeLocation string) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';

# Load data into table.
LOAD DATA INPATH 'asv://sanfrancrime@brunoterkaly.blob.core.windows.net/
SFPD_Incidents.csv' OVERWRITE INTO TABLE sfpdcrime;
select count(*) from sfpdcrime;

# Ad hoc query to aggregate and summarize crime types.
SELECT Descript, COUNT(*) AS cnt FROM sfpdcrime GROUP BY Descript
order by cnt desc;
```

advantage of these queries is that you never have to delve into Java and modify MapReduce functions, perform the compilation and packaging, or kick off the Hadoop job with the .jar file. Although you can remote in to the head node of the Hadoop cluster and perform these tasks (writing Java code, compiling the Java code, packaging it up as a .jar file, and using the .jar file to execute it as a Hadoop job), this isn't the optimal approach for most Hadoop users—it's too low-level.

The most productive way to run MapReduce jobs is to leverage the Windows Azure portal in HDInsight and issue Hive queries, assuming that using Pig is less technically appropriate. You can think of Hive as higher-level tooling that abstracts away the complexity of writing MapReduce functions in Java. It's really nothing more than a SQL-like scripting language. Queries written in Hive get compiled into Java MapReduce functions. Moreover, because Microsoft has contributed significant portions of optimization code for Hive in the Apache Hadoop project, chances are that queries written in Hive will be better optimized and will run more efficiently than hand-crafted code in Java. You can find an excellent tutorial at bit.ly/Wz1fbf.

The Hadoop Ecosystem

Once you leave the low-level world of writing MapReduce jobs in Java, you'll discover an incredible, highly evolved ecosystem of tooling that greatly extends Hadoop's capabilities. For example, Cloudera and Hortonworks are successful companies with business models based on Hadoop products, education and consulting services. Many open source projects provide additional capabilities, such as machine learning (ML); SQL-like query engines that support data summarization and ad hoc querying (Hive); data-flow language support (Pig); and much more. Here are just some of the projects that are worth a look: Sqoop, Pig, Apache Mahout, Cascading and Oozie. Microsoft offers a variety of tools as well, such as Excel with PowerPivot, Power View, and ODBC drivers that make it possible for Windows applications to issue queries against Hive data. Visit bit.ly/WleBeq to see a fascinating visual of the Hadoop ecosystem.

All of the Java and script code we presented previously can be replaced with the tiny amount of code in **Figure 5**. It's remarkable how three lines of code in Hive can efficiently achieve the same or better results than that previous code.

The most productive way
to run MapReduce jobs is to
leverage the Windows Azure
portal in HDInsight and issue
Hive queries.

There are some important points to note about **Figure 5**. First, notice that these commands look like familiar SQL statements, allowing you to create table structures into which you can load data. What's particularly interesting is the loading of data from Windows Azure Storage services. Note the asv prefix in the load statement in **Figure 5**. ASV stands for Azure Storage Vault, which you can use as a storage mechanism to provide input data to Hadoop jobs. As you may recall, while provisioning the process of an HDInsight cluster, you specified one or more specific Windows Azure Storage service accounts. The ability to leverage Windows Azure Storage services in HDInsight dramatically improves the usability and efficiency of managing and executing Hadoop jobs.

We've only scratched the surface in this article. There's a significant amount of sophisticated tooling that supports and extends HDInsight, and a variety of other open source projects you can learn about at the Apache Hadoop portal (hadoop.apache.org). Your next steps should include watching the Channel 9 video "Make Your Apps Smarter with Azure HDInsight" at bit.ly/190Vzfr. If your goal is to remain competitive by making decisions based on real data and analytics, HDInsight is there to help. ■

BRUNO TERKALY is a developer evangelist for Microsoft. His depth of knowledge comes from years of experience in the field, writing code using a multitude of platforms, languages, frameworks, SDKs, libraries and APIs. He spends time writing code, blogging and giving live presentations on building cloud-based applications, specifically using the Windows Azure platform. You can read his blog at blogs.msdn.com/b/brunoterkaly.

RICARDO VILLALOBOS is a seasoned software architect with more than 15 years of experience designing and creating applications for companies in the supply chain management industry. Holding different technical certifications, as well as a master's degree in business administration from the University of Dallas, he works as a cloud architect in the Windows Azure CSV incubation group for Microsoft. You can read his blog at blog.ricardovillalobos.com.

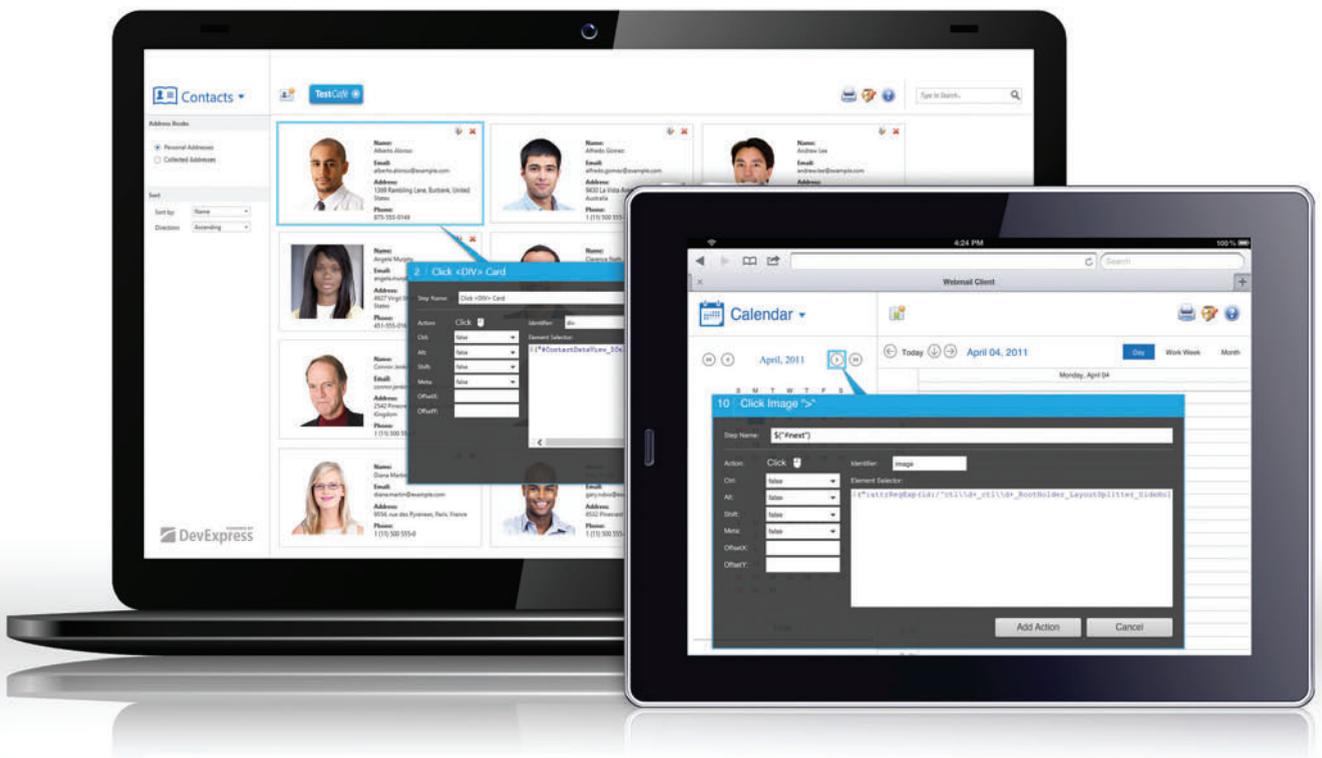
Terkaly and Villalobos jointly present at large industry conferences. They encourage readers of Windows Azure Insider to contact them for availability. Terkaly can be reached at bterkaly@microsoft.com and Villalobos can be reached at Ricardo.Villalobos@microsoft.com.

THANKS to the following technical experts for reviewing this article:
Paco Nathan (Concurrent Inc.) and Matt Winkler (Microsoft)

Introducing TestCafé

Affordable and Easy-to-Use Web Testing Framework

Record and execute tests on Windows®, Mac and Linux in any desktop or mobile browser that supports HTML5



Engineered to test any web application, anywhere on any device, TestCafé empowers developers and test engineers with industry-first capabilities designed to fundamentally improve the web testing experience and connected quality assurance processes.

To learn more and to experience TestCafé for yourself, visit TestCafe.DevExpress.com



Writing a Cross-Platform Presentation Layer with MVVM

Brent Edwards

With the release of Windows 8 and Windows Phone 8, Microsoft took a big step toward true cross-platform development. Both run on the same kernel now, which means with a little bit of planning, much of your application code can be reused in both. By leveraging the Model-View-ViewModel (MVVM) pattern, some other common design patterns, and a few tricks, you can write a cross-platform presentation layer that will work on both Windows 8 and Windows Phone 8.

In this article, I'll take a look at some specific cross-platform challenges I faced and talk about solutions that can be applied that allow my app to maintain clean separation of concerns without sacrificing the ability to write good unit tests for it.

This article discusses:

- The structure of the cross-platform solution
- Differences between the Windows 8 and Windows Phone 8 versions of the app
- Overcoming challenges related to navigation, application settings and secondary tiles

Technologies discussed:

Windows 8, Windows Phone 8, Visual Studio 2012

Code download available at:

archive.msdn.microsoft.com/mag201309MVVM

About the Sample App

In the July 2013 issue of *MSDN Magazine*, I presented code from a sample Windows Store application, and the start of an open source, cross-platform framework I developed, called Charmed (“Leveraging Windows 8 Features with MVVM,” msdn.microsoft.com/magazine/dn296512). In this article, I'll show you how I took that sample application and framework and evolved them to be more cross-platform. I also developed a companion Windows Phone 8 app with the same base functionality, leveraging the same framework. The framework and sample apps are available on GitHub at github.com/brentedwards/Charmed. The code will continue to evolve as I move toward the final article in my MVVM series, which will delve into actually testing the presentation layer and additional considerations around testable code.

The app is a simple blog reader, called Charmed Reader. Each platform's version of the app has just enough functionality to illustrate some key concepts relating to cross-platform development. Both versions are similar in the UX they provide, but still fit the look and feel of their respective OS.

Solution Structure

Any proper discussion of cross-platform development with Visual Studio 2012 must start at the beginning: solution structure. While Windows 8 and Windows Phone 8 do run on the same kernel, their apps still compile differently and have different project types. There are various ways to approach creating a solution with different project

types, but I prefer to have one solution with all my platform-specific projects included in it. **Figure 1** shows the solution structure for the sample apps I'll be discussing.

Visual Studio lets you have more than one project file reference a single physical class file, allowing you to add an existing class file by selecting Add As Link, as shown in **Figure 2**.

By leveraging the Add As Link functionality, I can write much of my code one time and use it for both Windows 8 and Windows Phone 8. However, I don't want to do this for every class file. As I'll demonstrate, there are situations where each platform will have its own implementation.

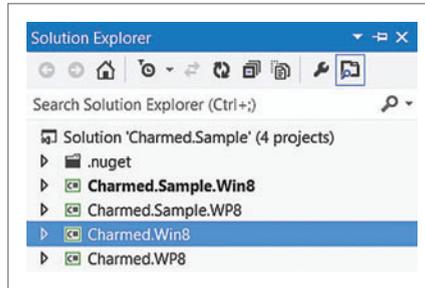


Figure 1 Cross-Platform Charmed Reader Solution Structure

uses 320 pixels because mobile designers have been designing for screen widths of 320 pixels for years. In cross-platform development, this works in my favor because I don't have to come up with a brand-new design for Snap View—I can just adapt my Windows Phone design. Of course, I have to keep in mind that each platform does have its own unique design principles, so I may have to vary a little to make each app feel natural to its platform.

As **Figure 3** shows, I implemented the UI for Windows 8 Snap View to be very similar but not quite identical to the UI for Windows Phone 8. Of course, I'm not a designer, and it shows in my thoroughly uninteresting UI design. But I hope this illustrates how similar Windows 8 Snap View and Windows Phone 8 can be in their UI designs.

Windows 8 and Windows Phone 8 have slightly different flavors of XAML that don't play well enough with each other to be interchangeable.

View Differences

Although I'll be able to reuse much of the code containing my presentation logic, which is written in C#, I won't be able to reuse the actual presentation code, which is written in XAML. This is because Windows 8 and Windows Phone 8 have slightly different flavors of XAML that don't play well enough with each other to be interchangeable. Part of the problem is syntax, particularly the namespace declarations, but most of it is due to different controls being available for the platforms as well as different styling concepts being implemented. For example, Windows 8 uses GridView and ListView quite heavily, but these controls aren't available for Windows Phone 8. On the flip side, Windows Phone 8 has the Pivot control and the LongListSelector, which aren't available in Windows 8.

Despite this lack of reusability of the XAML code, it's still possible to use some design assets in both platforms, particularly the Windows Phone UI design. This is because Windows 8 has the concept of Snap View, which has a fixed width of 320 pixels. Snap View

Code Differences

One of the interesting challenges I faced as I embarked on cross-platform development with Windows 8 and Windows Phone 8 is that each platform handles certain tasks differently. For example, while both platforms follow a URI-based navigation scheme, they vary in the parameters they take. Creating secondary tiles is also different. Though both platforms support them, what happens on each platform when a secondary tile is tapped is fundamentally different. Each platform also has its own way of dealing with application settings, and they have different classes for interacting with these settings.

Finally, there are features Windows 8 has that Windows Phone 8 doesn't. The main concepts my Windows 8 sample app leverages that Windows Phone 8 doesn't support are contracts and the Charms menu. This means that Windows Phone doesn't support the Share or Settings charms.

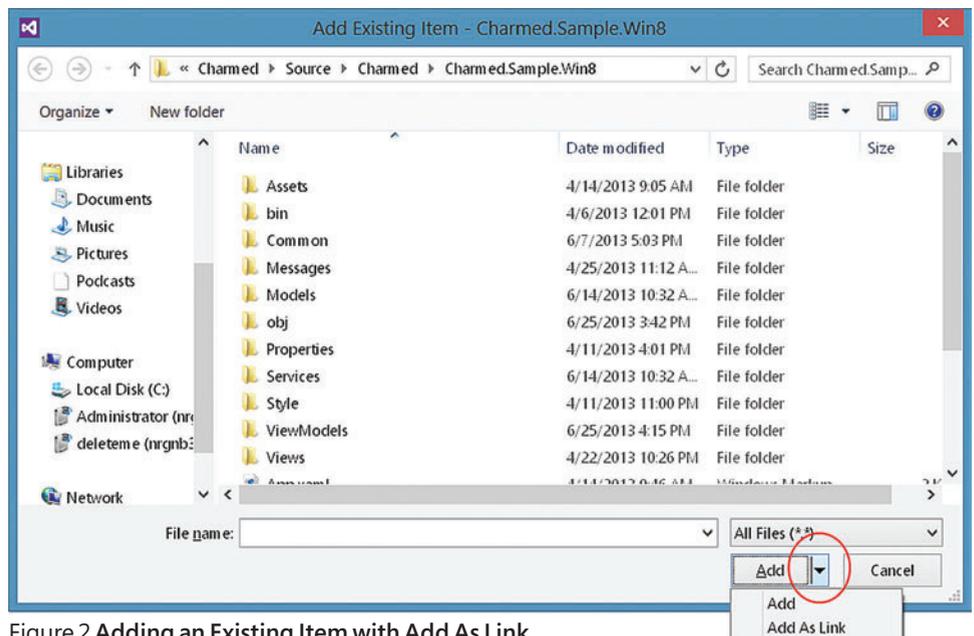


Figure 2 Adding an Existing Item with Add As Link

So, how do you deal with these fundamental code differences? There are a number of techniques you can employ to do so.

Compiler Directives When Visual Studio creates Windows 8 and Windows Phone 8 project types, it automatically defines platform-specific compiler directives in the project settings—NETFX_CORE for Windows 8 and WINDOWS_PHONE for Windows Phone 8. By leveraging these compiler directives, you can tell Visual Studio what to compile for each platform. Of the techniques that can be employed, this is the most basic, but it's also the messiest. It results in code that's a little like Swiss cheese: full of holes. While this is sometimes a necessary evil, there are better techniques that work in a lot of cases.

Abstraction This is the cleanest technique I use to deal with platform differences, and it involves abstracting the platform-specific functionality into an interface or an abstract class. This technique allows you to provide platform-specific implementations for interfaces and, at the same time, a consistent interface for use throughout the codebase. In cases where there's helper code each platform-specific implementation can utilize, you can implement an abstract class with this common helper code, then provide platform-specific implementations. This technique requires that the interface or abstract class be available in both projects, via the Add As Link functionality mentioned earlier.

Abstraction Plus Compiler Directives The final technique that can be used is a combination of the previous two. You can abstract the platform differences into an interface or an abstract class, then leverage compiler directives in the actual implementation. This is handy for cases where the platform differences are minor enough that it's not worth separating them for each project type.

By leveraging the
Add As Link functionality, I can
write much of my code one time
and use it for both Windows 8
and Windows Phone 8.

In practice, I've found that I rarely use compiler directives on their own, especially in my view models. I prefer to keep my view models clean whenever possible. So, when compiler directives are the best solution, I usually also slip in some abstraction to keep the Swiss cheese a little more hidden.

Navigation

One of the first challenges I encountered in my cross-platform travels was navigation. Navigation isn't the same in Windows 8 and Windows Phone 8, but it's close. Windows 8 now uses URI-based navigation, which Windows Phone has been using all along. The difference lies in how parameters are passed. Windows 8 takes a single object as a parameter while Windows Phone 8 takes as many

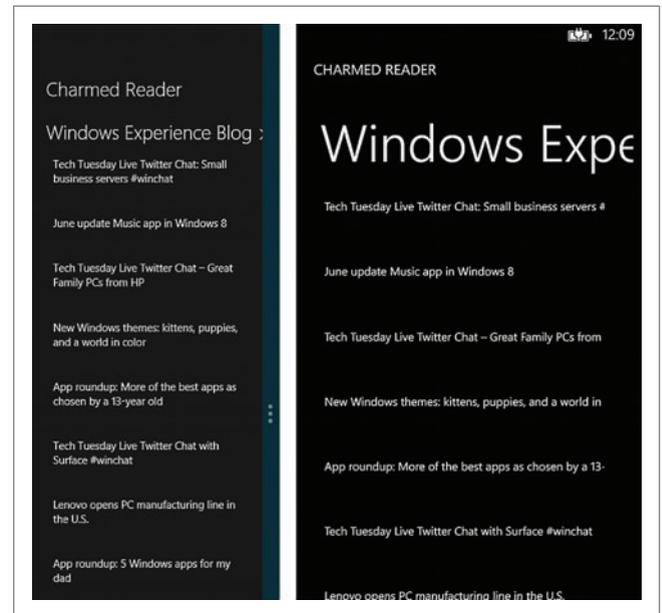


Figure 3 Sample App UI for Windows 8 Snap View (left) and Windows Phone 8 (right)

parameters as you like, but through the query string. Because Windows Phone uses the query string, all parameters must be serialized to a string. As it turns out, Windows 8 isn't so different in this respect.

Although Windows 8 takes a single object as a parameter, that object must somehow be serialized when another app takes center stage and my app gets deactivated. The OS takes the easy route and calls ToString on that parameter, which isn't all that helpful to me when my app gets activated again. Because I want to bring these two platforms together as much as possible in terms of my development effort, it makes sense to serialize my parameter as a string prior to navigation, then deserialize it after navigation is complete. I can even help facilitate this process with the implementation of my navigator.

Note that I want to avoid directly referencing the views from my view models, so I want the navigation to be view model-driven. My solution is to employ a convention in which views are placed in a Views namespace/folder and view models are placed in a ViewModels namespace/folder. I'll also make sure my views are named {Something}Page and my view models are named {Something}ViewModel. With this convention in place, I can provide some simple logic to resolve an instance of a view, based on the type of view model.

Now I need to decide what other functionality I need for navigation:

- View model-driven navigation
- The ability to go back
- For Windows Phone 8, the ability to remove a back stack entry

The first two are straightforward. I'll explain the need to remove a back stack entry a little later, but this ability is built in to Windows Phone 8 and not Windows 8.

Both Windows 8 and Windows Phone 8 leverage classes for their navigation that aren't easily mocked. Because one of my main goals with these apps is to keep them testable, I want to abstract this code behind a mockable interface. Therefore, my navigator will employ



The world's leading Imaging SDK
RUNS ANYWHERE



Document

OCR, Barcode & Forms Recognition

PDF Read, Write & Edit

Cleanup and Preprocessing



Medical

DICOM

PACS

Medical Workstation



Multimedia

Playback, Capture & Conversion

MPEG-2 Transport Stream

DVR



Imaging

Viewers

Image processing

150+ Formats

C++ C# JavaScript VB Objective-C Java

.NET Windows API WinRT Linux iOS OS X Android HTML5

DOWNLOAD OUR 60 DAY EVALUATION
WWW.LEADTOOLS.COM



SALES@LEADTOOLS.COM
 800.637.1840



a combination of abstraction and compiler directives. This leaves me with the following interface:

```
public interface INavigator
{
    bool CanGoBack { get; }
    void GoBack();
    void NavigateToViewModel<TViewModel>(object parameter = null);

#if WINDOWS_PHONE
    void RemoveBackEntry();
#endif // WINDOWS_PHONE
}
```

Note the use of #if WINDOWS_PHONE. This tells the compiler to compile the RemoveBackEntry into the interface definition only when the WINDOWS_PHONE compiler directive is defined, as it will be for Windows Phone 8 projects. Now here's my implementation, as shown in Figure 4.

I need to highlight a couple parts of the Navigator implementation in Figure 4, in particular the use of both the WINDOWS_PHONE

and NETFX_CORE compiler directives. This allows me keep the platform-specific code separated within the same code file. I also want to point out the ResolveViewUri method, especially how the query string parameter is defined. To keep things as consistent as possible across the two platforms, I'm allowing only one parameter to be passed. That one parameter will then be serialized and passed along in the platform-specific navigation. In the case of Windows Phone 8, that parameter will be passed via a "parameter" variable in the query string.

Of course, my implementation of navigator is pretty limited, particularly due to the overly simple convention expectation. If you're using an MVVM library, such as Caliburn.Micro, it can handle the actual navigation for you in a more robust way. However, in your own navigation, you may still want to apply this abstraction-plus-compiler-directives technique to smooth over the platform differences that exist in the libraries themselves.

Figure 4 Implementing INavigator

```
public sealed class Navigator : INavigator
{
    private readonly ISerializer serializer;
    private readonly IContainer container;

#if WINDOWS_PHONE
    private readonly Microsoft.Phone.Controls.PhoneApplicationFrame frame;
#endif // WINDOWS_PHONE

    public Navigator(
        ISerializer serializer,
        IContainer container
#if WINDOWS_PHONE
        , Microsoft.Phone.Controls.PhoneApplicationFrame frame
#endif // WINDOWS_PHONE
    )
    {
        this.serializer = serializer;
        this.container = container;

#if WINDOWS_PHONE
        this.frame = frame;
#endif // WINDOWS_PHONE
    }

    public void NavigateToViewModel<TViewModel>(object parameter = null)
    {
        var viewType = ResolveViewType<TViewModel>();

#if NETFX_CORE
        var frame = (Frame)Window.Current.Content;
#endif // NETFX_CORE

        if (parameter != null)
        {
#if WINDOWS_PHONE
            this.frame.Navigate(ResolveViewUri(viewType, parameter));
#else
            frame.Navigate(viewType, this.serializer.Serialize(parameter));
#endif // WINDOWS_PHONE
        }
        else
        {
#if WINDOWS_PHONE
            this.frame.Navigate(ResolveViewUri(viewType));
#else
            frame.Navigate(viewType);
#endif // WINDOWS_PHONE
        }
    }

    public void GoBack()
    {
#if WINDOWS_PHONE
        this.frame.GoBack();
#else
        ((Frame)Window.Current.Content).GoBack();
#endif // WINDOWS_PHONE
    }

    public bool CanGoBack
    {
        get
        {
#if WINDOWS_PHONE
            return this.frame.CanGoBack;
#else
            return ((Frame)Window.Current.Content).CanGoBack;
#endif // WINDOWS_PHONE
        }
    }

    private static Type ResolveViewType<TViewModel>()
    {
        var viewModelType = typeof(TViewModel);

        var viewName = viewModelType.AssemblyQualifiedName.Replace(
            viewModelType.Name,
            viewModelType.Name.Replace("ViewModel", "Page"));

        return Type.GetType(viewName.Replace("Model", string.Empty));
    }

    private Uri ResolveViewUri(Type viewType, object parameter = null)
    {
        var queryString = string.Empty;
        if (parameter != null)
        {
            var serializedParameter = this.serializer.Serialize(parameter);
            queryString = string.Format("?parameter={0}", serializedParameter);
        }

        var match = System.Text.RegularExpressions.Regex.Match(
            viewType.FullName, @"\s*.Views.*");
        if (match == null || match.Captures.Count == 0)
        {
            throw new ArgumentException("Views must exist in Views namespace.");
        }
        var path = match.Captures[0].Value.Replace('.', '/');

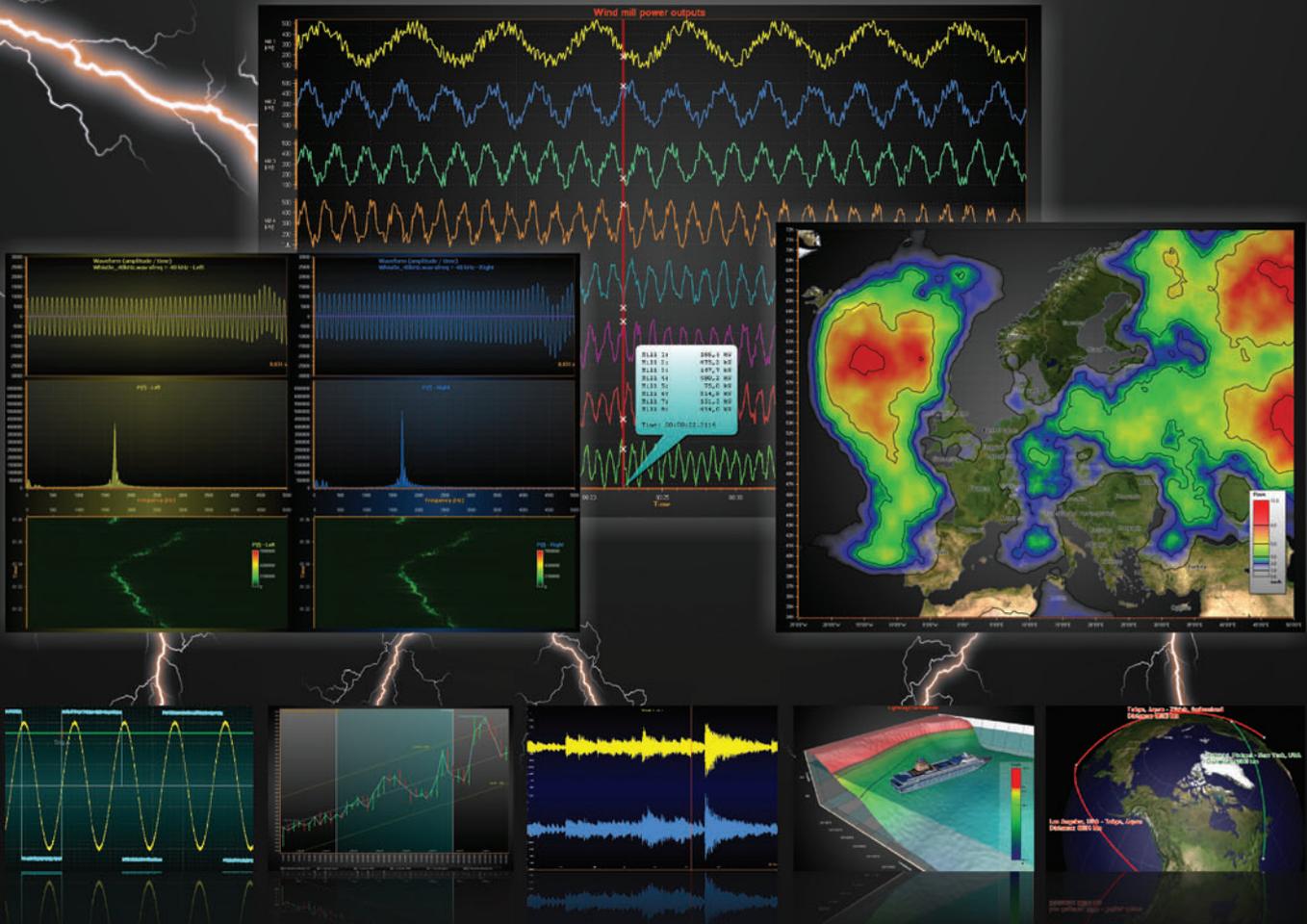
        return new Uri(string.Format("{0}.xaml[1]", path, queryString),
            UriKind.Relative);
    }

#if WINDOWS_PHONE
    public void RemoveBackEntry()
    {
        this.frame.RemoveBackEntry();
    }
#endif // WINDOWS_PHONE
}
```

ULTIMATE CHARTING POWER

LightningChart

The fastest rendering data visualization components
for WPF and WinForms



- Fully DirectX accelerated
- Superior 2D and 3D rendering performance
- Very extensive property sets
- Optimized for real-time data monitoring
- Supports gigantic data sets
- Professional, friendly and fast customer support
- Compatible with Visual Studio 2005...2012

WPF charts performance comparison

Opening large dataset	LightningChart is up to 977,000 % faster
Real-time monitoring	LightningChart is up to 2,700,000 % faster

Winforms charts performance comparison

Opening large dataset	LightningChart is up to 37,000 % faster
Real-time monitoring	LightningChart is up to 2,300,000 % faster

Results compared to average of other chart controls. See details at www.LightningChart.com/benchmark. LightningChart results apply for Ultimate edition.

FREE TRIAL

Download a free 30-day evaluation from
www.LightningChart.com

Arction
Pioneers of high-performance data visualization

Application Settings

Application settings are another area where Windows 8 and Windows Phone 8 diverge. Each platform has the ability to save application settings fairly easily, and their implementations are quite similar as well. They differ in the classes they use, but both use classes that aren't easily mockable, which would break the testability of my view model. So, once again, I'm going to opt for abstraction plus compiler directives. I must first decide what my interface should look like. The interface must:

- Add or update a setting
- Try to get a setting, without throwing an exception on failure
- Remove a setting
- Determine if a setting exists for a given key

Pretty straightforward stuff, so my interface will be pretty straightforward:

```
public interface ISettings
{
    void AddOrUpdate(string key, object value);
    bool TryGetValue<T>(string key, out T value);
    bool Remove(string key);
    bool ContainsKey(string key);
}
```

Because both platforms will have the same functionality, I don't have to bother with compiler directives in the interface, keeping things nice and clean for my view models. **Figure 5** shows my implementation of the `ISettings` interface.

The implementation shown in **Figure 5** is pretty straightforward, just like the `ISettings` interface itself. It illustrates how each platform is only slightly different and requires only slightly different code to add, retrieve and remove application settings. The one thing I will point out is that the Windows 8 version of the code leverages roaming settings—Windows 8-specific functionality that lets an app store its settings in the cloud, allowing users to open the app on another Windows 8 device and see the same settings applied.

Both Windows 8 and Windows Phone 8 support the creation of secondary tiles.

Secondary Tiles

As I said earlier, both Windows 8 and Windows Phone 8 support the creation of secondary tiles—tiles that are created programmatically and pinned to the user's home screen. Secondary tiles provide deep-linking functionality, which means the user can tap a secondary tile and jump straight to a specific part of an application. This is valuable for users because they can essentially bookmark parts of your app and jump right to them without wasting time. In the case of my sample apps, I want the user to be able to bookmark a single blog post (`FeedItem`) and jump right to it from the home screen.

The interesting thing with secondary tiles is that, while both platforms support them, the way I must implement them for each platform is quite different. This is a perfect case for exploring a more complex abstraction example.

If you read my July 2013 article, you may recall that I talked about how to abstract secondary tiles for MVVM development with Windows 8. The solution I presented works perfectly fine for Windows 8, but doesn't even compile for Windows Phone 8. What I'll present here is the natural evolution of that Windows 8 solution, which works well on both Windows 8 and Windows Phone 8.

Here's what the interface looked like in Windows 8:

```
public interface ISecondaryPinner
{
    Task<bool> Pin(FrameworkElement anchorElement,
        Placement requestPlacement, TileInfo tileInfo);
    Task<bool> Unpin(FrameworkElement anchorElement,
        Placement requestPlacement, string tileId);
    bool IsPinned(string tileId);
}
```

As I mentioned, this interface doesn't compile for Windows Phone 8. Particularly problematic is the use of `FrameworkElement` and the `Placement` enumeration. Neither of these are the same in

Figure 5 Implementing `ISettings`

```
public sealed class Settings : ISettings
{
    public void AddOrUpdate(string key, object value)
    {
        #if WINDOWS_PHONE
            IsolatedStorageSettings.ApplicationSettings[key] = value;
            IsolatedStorageSettings.ApplicationSettings.Save();
        #else
            ApplicationData.Current.RoamingSettings.Values[key] = value;
        #endif // WINDOWS_PHONE
    }

    public bool TryGetValue<T>(string key, out T value)
    {
        #if WINDOWS_PHONE
            return IsolatedStorageSettings.ApplicationSettings.TryGetValue<T>(
                key, out value);
        #else
            var result = false;
            if (ApplicationData.Current.RoamingSettings.Values.ContainsKey(key))
            {
                value = (T)ApplicationData.Current.RoamingSettings.Values[key];
                result = true;
            }
            else
            {
                value = default(T);
            }

            return result;
        #endif // WINDOWS_PHONE
    }

    public bool Remove(string key)
    {
        #if WINDOWS_PHONE
            var result = IsolatedStorageSettings.ApplicationSettings.Remove(key);
            IsolatedStorageSettings.ApplicationSettings.Save();
            return result;
        #else
            return ApplicationData.Current.RoamingSettings.Values.Remove(key);
        #endif // WINDOWS_PHONE
    }

    public bool ContainsKey(string key)
    {
        #if WINDOWS_PHONE
            return IsolatedStorageSettings.ApplicationSettings.ContainsKey(key);
        #else
            return ApplicationData.Current.RoamingSettings.Values.ContainsKey(key);
        #endif // WINDOWS_PHONE
    }
}
```

NEW HOSTING

DYNAMIC PRODUCTS, FLEXIBILITY, AND SUPPORT
FOR YOUR WEB PROJECTS



APPS

TOOLS

TECHNOLOGY

PACKAGES



1and1.com

NEW WHO

THE NEW PROFESSION

We are Internet and eBusiness experts, with more than 6,000 specialists in 10 countries. The Internet has been our business for over 25 years, our enthusiasm and know-how has resulted in state-of-the-art technology, 19 million hosted websites in 5 high-performance data centers, and over 12 million customer accounts. See why so many are choosing 1&1 as their web hosting partner.



DOMAINS | E-MAIL | WEB HOSTING | eCOMMERCE | SERVERS

STING

NAL STANDARD

We know what professionals want – flexible solutions that are perfectly matched to their needs.



APPS

TOOLS

TECHNOLOGY

PACKAGES



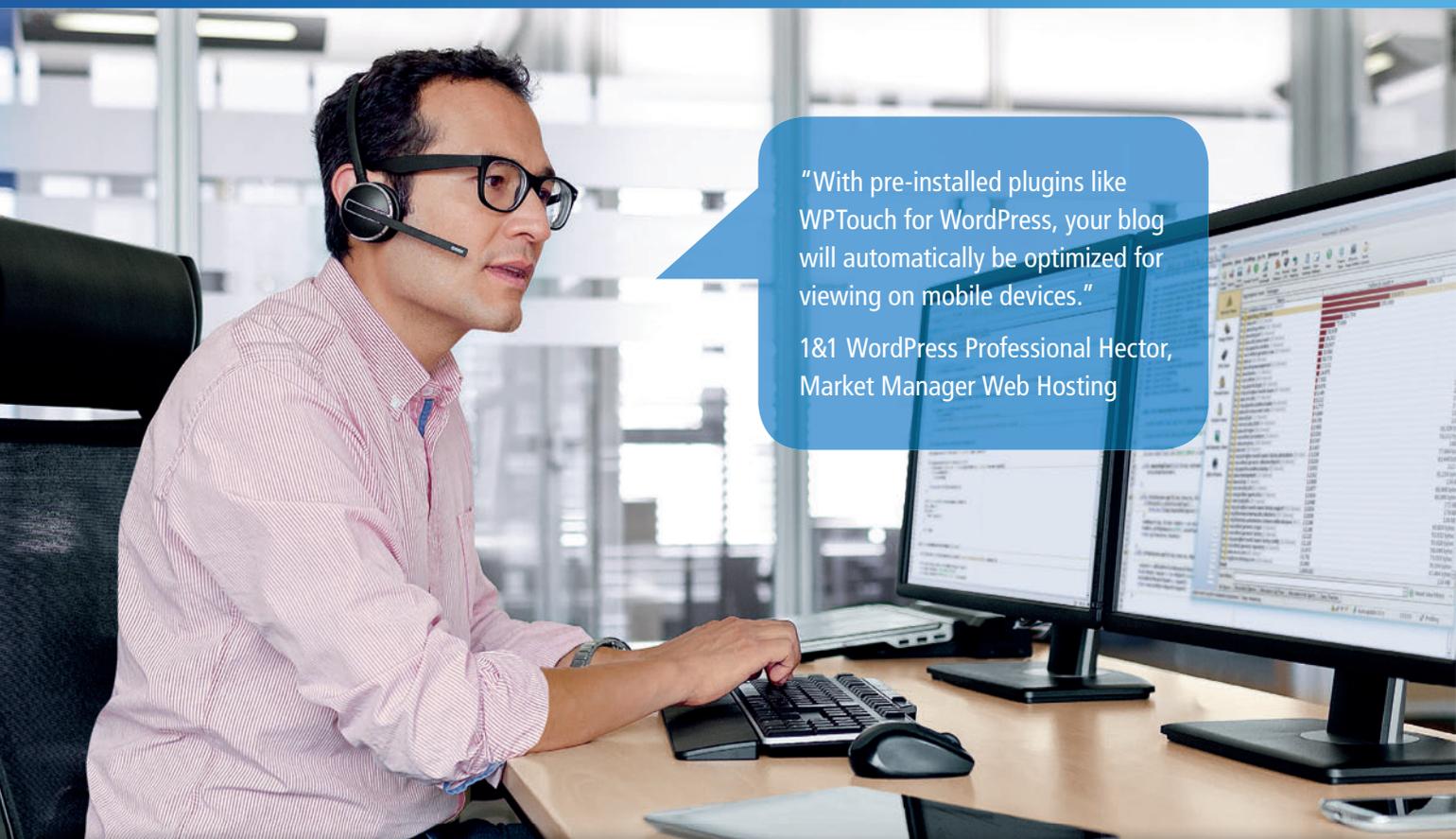
Call **1 (877) 461-2631** or buy online

1and1.com

THE NEW 1&1

APP

EXPERT CENTER



"With pre-installed plugins like WPTouch for WordPress, your blog will automatically be optimized for viewing on mobile devices."

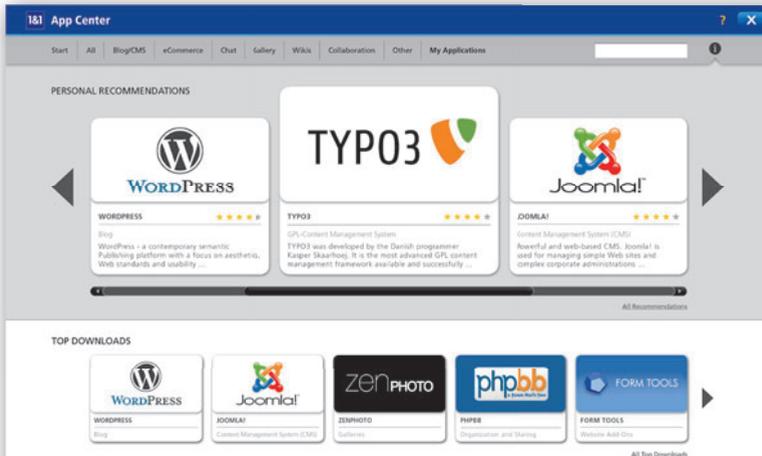
1&1 WordPress Professional Hector,
Market Manager Web Hosting

DOMAINS | E-MAIL | WEB HOSTING | eCOMMERCE | SERVERS

In our 1&1 App Expert Center, you will find the 40 most popular applications featuring useful plug-ins and one-click installation. Plus, get tips and tricks on how to get the most from each application.

EXPERTS ON TOP APPS

The 1&1 App Expert team provides support for the 5 most used apps and plug-ins, PHP and MySQL and support for every stage of your web project. Our experts will support you all the way!



YOU CHOOSE:

FREE MODE OR SAFE MODE?

If you want maximum freedom and control, then use Free Mode. You'll benefit from time-saving 1-click installation, as well as the ability to code plug-ins yourself at any time! We'll keep you informed regarding necessary updates and security patches so you can implement them yourself – in Free Mode, you're the boss!

If convenience and security are your top priorities, then use Safe Mode. While you'll only have access to the default plug-ins, we manage the apps for you, take care of all updates and security patches, and more. Plus, if you feel confident you always have the option of switching to Free Mode and personally taking control!

ALSO INCLUDED:

- **1&1 WEB APP PORTAL** with HTML code for over 100 popular widgets
- **GUARANTEED RESOURCES** for demanding applications



Call **1 (877) 461-2631** or buy online

1and1.com

PROFESSIONAL

TOOLS

FOR YOUR WEB PROJECTS

LATEST PHP VERSION

Do you regularly use PHP? With 1&1 you'll have access to PHP 5.4 from Zend, the component-based framework for PHP, with additional support from the 1&1 App Center Experts.



1&1 MOBILE WEBSITE BUILDER

1&1 Mobile Website Builder automatically converts your website for mobile optimization to be viewed on smartphones and tablets. You can also manually edit all of your pages.

DOMAINS | E-MAIL | WEB HOSTING | eCOMMERCE | SERVERS

"Need flexibility in your design software? NetObjects Fusion 2013 offers web design, image editing, database connectivity, FTP, CSS & HTML editors, backup, animation, task management and much more."

1&1 Web Design Professional Daniel,
Commercial Manager Web Hosting



SOFTWARE INCLUDED!



ADDITIONALLY:

- **SOFTWARE:** **git-versioning** available via SSH
- **LINUX:** Web space recovery with up to 6 restore points at anytime, support for additional languages including Perl, Python and Ruby
- **WINDOWS:** ASP.NET 4.5, ASP MVC 3 and 4, PHP and Perl support for .NET Framework, dedicated app pools, and up to 10 .NET Applications.



Call **1 (877) 461-2631** or buy online

1and1.com

STATE-OF-THE-ART

TECHNOL

The high-performance 1&1 Data Centers are among the safest and most efficient in the world. Redundant data centers provide the highest level of availability.



NEW CONTENT DELIVERY NETWORK

1&1 CDN (Content Delivery Network) provides maximum performance for your website. POPs are distributed over 23 locations worldwide on different backbones. Through this network, the data of static website pages is stored closer to your visitors. This enables your visitors to get fast access to your website.

>300 GBIT/S CONNECTION

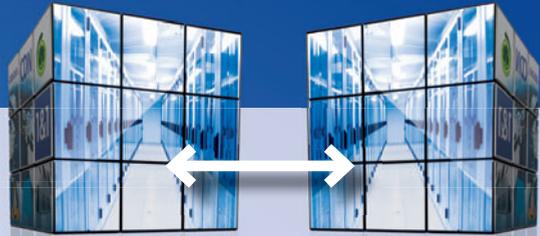
Our own external network connectivity backbone! To avoid data transfer congestion, 1&1 Data Centers are geo-redundant so that most important Internet nodes can be tethered, meaning that the fastest possible connection is automatically selected. You will also benefit from maximum uptime.

2 GB RAM GUARANTEED

Our top shared hosting package gives you the experience of a dedicated server – with 2 GB RAM for guaranteed performance! This makes demanding applications and large dynamic sites run smoothly with dedicated resources. Your site will also perform better during high volume visitor traffic.

DOMAINS | E-MAIL | WEB HOSTING | eCOMMERCE | SERVERS

OGY



"1&1 offers the highest standard of data security. With geo-redundancy, your data is stored simultaneously in two of our high-performance data centers in two separate locations. Our robust infrastructure ensures that your website offers the best possible performance."

1&1 Technology Specialist Stefan,
Head of Development Web Hosting & Servers



ALSO INCLUDES:

- **DAILY BACKUPS**
For the complete infrastructure
- **PROACTIVE MONITORING**
From 1&1 Experts
- **NEW SOFTWARE VERSIONS**
With the latest features and security patches



Call **1 (877) 461-2631** or buy online

1and1.com

1&1 NEW HOSTING

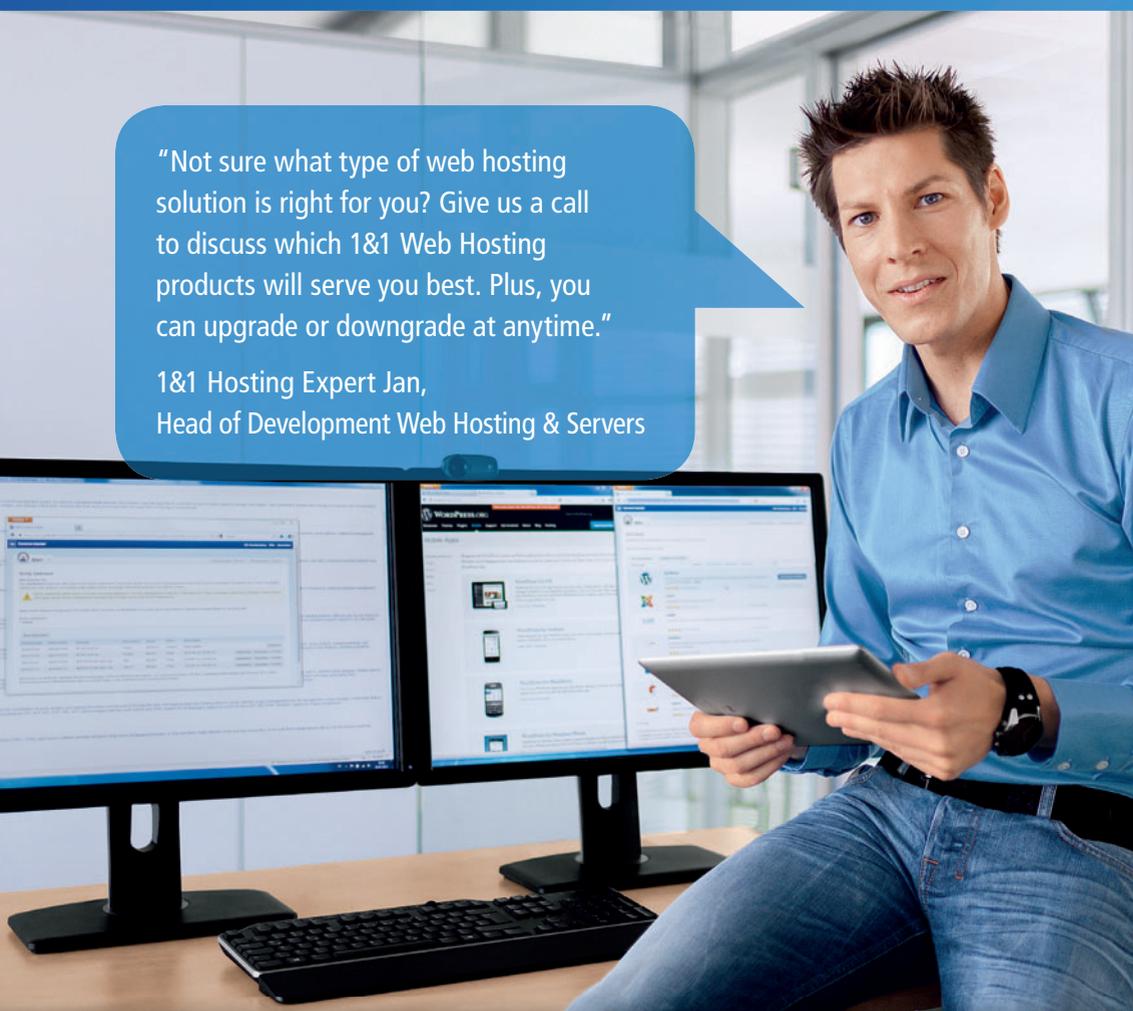
PACKAGES

TO FIT YOUR NEEDS

"Not sure what type of web hosting solution is right for you? Give us a call to discuss which 1&1 Web Hosting products will serve you best. Plus, you can upgrade or downgrade at anytime."

1&1 Hosting Expert Jan,
Head of Development Web Hosting & Servers

With 1&1, you'll always find the perfect package for your needs. Our website is constantly updated with current offers for virtual, cloud and dedicated servers, eCommerce and other eBusiness solutions.



DOMAINS | E-MAIL | WEB HOSTING | eCOMMERCE | SERVERS

* 1&1 Web Hosting packages come with a 30 day money back guarantee, no minimum contract term, and no setup fee. Prices in table reflect 36 month pre-payment option.



	Basic	Unlimited	Advanced
	One or more websites, high memory demands	Websites with interactive and dynamic content	Guaranteed performance, resource-intensive projects
Domains included (.com .net .org .biz .info)	1	1	1
No. of websites	1	UNLIMITED	UNLIMITED
Web space	100 GB	UNLIMITED	UNLIMITED
Traffic	UNLIMITED	UNLIMITED	UNLIMITED
E-mail accounts	100	UNLIMITED	UNLIMITED
Databases	20	UNLIMITED	UNLIMITED
APPS			
Click & Build Apps	✓	✓	✓
Web Apps	✓	✓	✓
Expert Support for top apps, PHP and MySQL	-	-	✓
Guaranteed Resources	-	-	2 GB RAM
TOOLS			
1&1 Mobile Website Builder	-	✓	✓
Premium software	NetObjects® Fusion® 2013 1&1 Edition	NetObjects® Fusion® 2013 1&1 Edition	NetObjects® Fusion® 2013, Adobe® Dreamweaver CS 5.5
PHP 5.4	✓	✓	✓
Git version control tool	✓	✓	✓
Perl, Python, Ruby	✓	✓	✓
INFRASTRUCTURE			
Maximum availability (Geo-redundancy)	✓	✓	✓
Redundant Network connectivity	> 300 Gbit/s	> 300 Gbit/s	> 300 Gbit/s
1&1 CDN powered by CloudFlare	-	✓	✓
1&1 Webspaces Recovery	✓	✓	✓
Support	24/7	24/7	24/7 plus App Expert Support
	\$1.99 \$5.99 per month*	\$3.99 \$8.99 per month*	\$5.99 \$14.99 per month*



Call **1 (877) 461-2631** or buy online

1and1.com

THE 1&1 PRINCIPLES

THE PROFESSIONAL DIFFERENCE



TRIAL

... with 1&1 you can test your package knowing you have a 30 day money back guarantee. If you are not satisfied for any reason you can easily cancel to receive a refund.

MONTH

... short term payment options on request. If you don't want to commit for more than one month, then choose maximum flexibility with monthly billing, or save more with a longer pre-paid plan.

CLICK

... with just the click of a button you can upgrade or downgrade your hosting package. Change your package within the same product group once a month at no extra cost and only pay for what you use.

CALL

... all it takes is one call to speak with an expert 24/7. Our experts are always here for you.

POSITIVE OUTCOME

... 1&1 Data Centers are state-of-the-art. We use geo-redundancy so that in the unlikely event of a failure, fast switching between our data centers provide maximum reliability ... another reason millions trust 1&1.



Call **1 (877) 461-2631** or buy online

1and1.com

Figure 6 Implementing ISecondaryPinner for Windows 8

```
public sealed class Win8SecondaryPinner : ISecondaryPinner
{
    public async Task<bool> Pin(TileInfo tileInfo)
    {
        if (tileInfo == null)
        {
            throw new ArgumentNullException("tileInfo");
        }

        var isPinned = false;

        if (!SecondaryTile.Exists(tileInfo.TileId))
        {
            var secondaryTile = new SecondaryTile(
                tileInfo.TileId,
                tileInfo.ShortName,
                tileInfo.DisplayName,
                tileInfo.Arguments,
                tileInfo.TileOptions,
                tileInfo.LogoUri);

            if (tileInfo.WideLogoUri != null)
            {
                secondaryTile.WideLogo = tileInfo.WideLogoUri;
            }

            isPinned = await secondaryTile.RequestCreateForSelectionAsync(
                GetElementRect(tileInfo.AnchorElement), tileInfo.RequestPlacement);
        }

        return isPinned;
    }

    public async Task<bool> Unpin(TileInfo tileInfo)
    {
        var wasUnpinned = false;

        if (SecondaryTile.Exists(tileInfo.TileId))
        {
            var secondaryTile = new SecondaryTile(tileInfo.TileId);
            wasUnpinned = await secondaryTile.RequestDeleteForSelectionAsync(
                GetElementRect(tileInfo.AnchorElement), tileInfo.RequestPlacement);
        }

        return wasUnpinned;
    }

    public bool IsPinned(string tileId)
    {
        return SecondaryTile.Exists(tileId);
    }

    private static Rect GetElementRect(FrameworkElement element)
    {
        GeneralTransform buttonTransform = element.TransformToVisual(null);
        Point point = buttonTransform.TransformPoint(new Point());
        return new Rect(point, new Size(element.ActualWidth, element.ActualHeight));
    }
}
```

Windows Phone 8 as in Windows 8. My goal is to modify this interface a little so it can be used by both platforms without issue. As you can see, the `ISecondaryPinner.Pin` method takes as a parameter a `TileInfo` object. `TileInfo` is a simple data transfer object (DTO) I created containing relevant information needed for creating a secondary tile. The easiest thing for me to do is to move the parameters the Windows 8 version needs into the `TileInfo` class, then use compiler directives to compile them into the Windows 8 version of the `TileInfo` class. Doing so changes my `ISecondaryPinner` interface to the following:

```
public interface ISecondaryPinner
{
    Task<bool> Pin(TileInfo tileInfo);
    Task<bool> Unpin(TileInfo tileInfo);
    bool IsPinned(string tileId);
}
```

You can see that the methods are all still the same, but the parameters for `Pin` and `Unpin` have changed slightly. As a result, the `TileInfo` class has also changed from what it was previously and now looks like this:

```
public sealed class TileInfo
{
    public string TileId { get; set; }
    public string ShortName { get; set; }
    public string DisplayName { get; set; }
    public string Arguments { get; set; }
    public Uri LogoUri { get; set; }
    public Uri WideLogoUri { get; set; }

    public string AppName { get; set; }
    public int? Count { get; set; }

    #if NETFX_CORE
    public Windows.UI.StartScreen.TileOptions TileOptions { get; set; }
    public Windows.UI.Xaml.FrameworkElement AnchorElement { get; set; }
    public Placement RequestPlacement { get; set; }
    #endif // NETFX_CORE
}
```

In reality, I prefer to provide constructors for each scenario in which helper DTOs like this get used, in order to be more explicit about what parameters are needed at what times. For the sake of brevity, I left the different constructors out of the `TileInfo` code snippet, but you can see them in all their glory in the sample code.

Figure 7 Implementing ISecondaryPinner for Windows Phone 8

```
public sealed class WP8SecondaryPinner : ISecondaryPinner
{
    public Task<bool> Pin(TileInfo tileInfo)
    {
        var result = false;
        if (!this.IsPinned(tileInfo.TileId))
        {
            var tileData = new StandardTileData
            {
                Title = tileInfo.DisplayName,
                BackgroundImage = tileInfo.LogoUri,
                Count = tileInfo.Count,
                BackTitle = tileInfo.AppName,
                BackBackgroundImage = new Uri("", UriKind.Relative),
                BackContent = tileInfo.DisplayName
            };

            ShellTile.Create(new Uri(tileInfo.TileId, UriKind.Relative), tileData);
            result = true;
        }

        return Task.FromResult<bool>(result);
    }

    public Task<bool> Unpin(TileInfo tileInfo)
    {
        ShellTile tile = this.FindTile(tileInfo.TileId);
        if (tile != null)
        {
            tile.Delete();
        }

        return Task.FromResult<bool>(true);
    }

    public bool IsPinned(string tileId)
    {
        return FindTile(tileId) != null;
    }

    private ShellTile FindTile(string uri)
    {
        return ShellTile.ActiveTiles.FirstOrDefault(
            tile => tile.NavigationUri.ToString() == uri);
    }
}
```

TileInfo now has all the properties needed by both Windows 8 and Windows Phone 8, so the next thing to do is implement the *ISecundaryPinner* interface. Because the implementation will be quite different for each platform, I'll use the same interface in both project types but provide platform-specific implementations in each project. This will cut down on the Swiss-cheese effect that

Figure 8 FeedItemViewModel Base Class

```
public abstract class FeedItemViewModel : ViewModelBase<FeedItem>
{
    private readonly IStorage storage;
    protected readonly ISecundaryPinner secundaryPinner;

    public FeedItemViewModel(
        ISerializer serializer,
        IStorage storage,
        ISecundaryPinner secundaryPinner)
        : base(serializer)
    {
        this.storage = storage;
        this.sekundaryPinner = secundaryPinner;
    }

    public override void LoadState(FeedItem navigationParameter,
        Dictionary<string, object> pageState)
    {
        this.FeedItem = navigationParameter;
    }

    protected async Task SavePinnedFeedItem()
    {
        var pinnedFeedItems =
            await this.storage.LoadAsync<List<FeedItem>>(
                Constants.PinnedFeedItemsKey);

        if (pinnedFeedItems == null)
        {
            pinnedFeedItems = new List<FeedItem>();
        }

        pinnedFeedItems.Add(feedItem);
        await this.storage.SaveAsync(Constants.PinnedFeedItemsKey, pinnedFeedItems);
    }

    protected async Task RemovePinnedFeedItem()
    {
        var pinnedFeedItems =
            await this.storage.LoadAsync<List<FeedItem>>(
                Constants.PinnedFeedItemsKey);

        if (pinnedFeedItems != null)
        {
            var pinnedFeedItem = pinnedFeedItems.FirstOrDefault(fi => fi.Id ==
                this.FeedItem.Id);
            if (pinnedFeedItem != null)
            {
                pinnedFeedItems.Remove(pinnedFeedItem);
            }

            await this.storage.SaveAsync(Constants.PinnedFeedItemsKey, pinnedFeedItems);
        }
    }

    private FeedItem feedItem;
    public FeedItem FeedItem
    {
        get { return this.feedItem; }
        set { this.SetProperty(ref this.feedItem, value); }
    }

    private bool isFeedItemPinned;
    public bool IsFeedItemPinned
    {
        get { return this.isFeedItemPinned; }
        set { this.SetProperty(ref this.isFeedItemPinned, value); }
    }
}
```

compiler directives would cause in this case. **Figure 6** shows the Windows 8 implementation of *ISecundaryPinner*, now with the updated method signatures.

It's important to note that in Windows 8 you can't quietly create a secondary tile programmatically without approval from the user. This is different from Windows Phone 8, which does let you do so. So, I must create an instance of *SecondaryTile* and call *RequestCreateForSelectionAsync*, which will pop up a dialog in the position I provide, prompting the user to approve the creation (or deletion) of the secondary tile. The helper method *GetElementRect* takes in a *FrameworkElement*—which will be the button the user presses to pin the secondary tile—and then calculates its rectangle to be used for positioning the request dialog.

Figure 7 shows the Windows Phone 8 implementation of *ISecundaryPinner*.

In Windows 8 you can't quietly create a secondary tile programmatically without approval from the user.

There are a couple of things I'd like to point out in the Windows Phone 8 implementation. The first is how the secondary tile is created using the *StandardTileData* class and the static *ShellTile.Create* method. The second is that the Windows Phone 8 implementation for creating secondary tiles is not asynchronous. Because the Windows 8 implementation *is* asynchronous, I had to make the interface support the *async/await* pattern. Luckily, it's very easy to make what would otherwise be a non-asynchronous method support the *async/await* pattern by leveraging the static, generic method *Task.FromResult*. The view models that use the *ISecundaryPinner* interface don't need to worry about the fact that Windows 8 is inherently asynchronous and Windows Phone 8 is not.

You can now see how Windows 8 (**Figure 6**) and Windows Phone 8 (**Figure 7**) vary in their implementation of secondary tiles. That's not the end of the story for secondary tiles, however. I've shown only the implementation of the *ISecundaryPinner* interface. Because each platform is different and must provide values for different properties of the *TileInfo* class, I must also provide platform-specific implementations of the view models that use them. For my sample application, I'll provide the ability to pin a single blog post, or *FeedItem*, so the view model in question is the *FeedItemViewModel*.

Some common functionality does exist in both Windows 8 and Windows Phone 8, from the view model's perspective. When the user pins a *FeedItem*, I want both platforms to save that *FeedItem* locally so it can be reloaded when the user taps its secondary tile. On the flip side, when the user unpins a *FeedItem*, I want both platforms to delete that *FeedItem* from local storage. Both platforms will need to implement this common functionality, yet expand on it with platform-specific implementations for the secondary tile functionality. So, it makes sense to provide a base class that

DEVELOPED FOR INTUITIVE USE

DynamicPDF—Comprehensive PDF Solutions for .NET Developers

ceTe Software's DynamicPDF products provide real-time PDF generation, manipulation, conversion, printing, viewing, and much more. Providing the best of both worlds, the object models are extremely flexible but still supply the rich features you need as a developer. Reliable and efficient, the high-performance software is easy to learn and use. If you do encounter a question with any of our components, simply contact ceTe Software's readily available, industry-leading support team.



DynamicPDF

WWW.DYNAMICPDF.COM



TRY OUR PDF SOLUTIONS FREE TODAY!

www.DynamicPDF.com/eval or call 800.631.5006 | +1 410.772.8620

ceTe software

Figure 9 Concrete Implementation of FeedItemViewModel for Windows 8

```

public sealed class Win8FeedItemViewModel : FeedItemViewModel
{
    private readonly IShareManager shareManager;

    public Win8FeedItemViewModel(
        ISerializer serializer,
        IStorage storage,
        ISecondaryPinner secondaryPinner,
        IShareManager shareManager)
        : base(serializer, storage, secondaryPinner)
    {
        this.shareManager = shareManager;
    }

    public override void LoadState(Models.FeedItem navigationParameter,
        Dictionary<string, object> pageState)
    {
        base.LoadState(navigationParameter, pageState);

        this.IsFeedItemPinned =
            this.secondaryPinner.IsPinned(FormatSecondaryTileId());
    }

    public override void SaveState(Dictionary<string, object> pageState)
    {
        base.SaveState(pageState);

        this.shareManager.Cleanup();
    }

    public async Task Pin(Windows.UI.Xaml.FrameworkElement anchorElement)
    {
        // Pin the feed item, then save it locally to make sure it is still
        // available when they return.
        var tileInfo = new TileInfo(
            this.FormatSecondaryTileId(),
            this.FeedItem.Title,
            this.FeedItem.Title,
            Windows.UI.StartScreen.TileOptions.ShowNameOnLogo |
            Windows.UI.StartScreen.TileOptions.ShowNameOnWideLogo,
            new Uri("ms-appx:///Assets/Logo.png"),
            new Uri("ms-appx:///Assets/WideLogo.png"),
            anchorElement,
            Windows.UI.Popups.Placement.Above,
            this.FeedItem.Id.ToString());

        this.IsFeedItemPinned = await this.secondaryPinner.Pin(tileInfo);

        if (this.IsFeedItemPinned)
        {
            await SavePinnedFeedItem();
        }
    }

    public async Task Unpin(Windows.UI.Xaml.FrameworkElement anchorElement)
    {
        // Unpin, then delete the feed item locally.
        var tileInfo = new TileInfo(this.FormatSecondaryTileId(), anchorElement,
            Windows.UI.Popups.Placement.Above);
        this.IsFeedItemPinned = !await this.secondaryPinner.Unpin(tileInfo);

        if (!this.IsFeedItemPinned)
        {
            await RemovePinnedFeedItem();
        }
    }

    private string FormatSecondaryTileId()
    {
        return string.Format(Constants.SecondaryIdFormat, this.FeedItem.Id);
    }
}

```

implements the common functionality and make that base class available on both platforms. Then, each platform can inherit that base class with platform-specific classes that provide the platform's specific implementations for pinning and unpinning secondary tiles.

Figure 8 shows the FeedItem ViewModel, which is the base class to be inherited by both platforms. FeedItem ViewModel contains all of the stuff that's common to both platforms.

With a base class in place to facilitate saving and deleting pinned FeedItems, I can move on to platform-specific implementations. The concrete implementation of FeedItem ViewModel for Windows 8

and the use of the TileInfo class with the properties that Windows 8 cares about is shown in Figure 9.

Figure 10 shows the concrete implementation of FeedItem-ViewModel for Windows Phone 8. Using TileInfo for Windows Phone 8 requires fewer properties than for Windows 8.

After the Windows 8 (Figure 9) and Windows Phone 8 (Figure 10) implementations for FeedItem ViewModel, my apps are all set to pin secondary tiles to their respective home screens. The only thing remaining to close the loop on the functionality is to handle when the user actually taps the pinned secondary tiles. My goal for

Figure 10 Concrete Implementation of FeedItemViewModel for Windows Phone 8

```

public sealed class WP8FeedItemViewModel : FeedItemViewModel
{
    public WP8FeedItemViewModel(
        ISerializer serializer,
        IStorage storage,
        ISecondaryPinner secondaryPinner)
        : base(serializer, storage, secondaryPinner)
    {
    }

    public async Task Pin()
    {
        // Pin the feed item, then save it locally to make sure it is still
        // available when they return.
        var tileInfo = new TileInfo(
            this.FormatTileIdUrl(),
            this.FeedItem.Title,
            Constants.AppName,
            new Uri("/Assets/ApplicationIcon.png", UriKind.Relative));

        this.IsFeedItemPinned = await this.secondaryPinner.Pin(tileInfo);

        if (this.IsFeedItemPinned)
        {
            await this.SavePinnedFeedItem();
        }
    }

    public async Task Unpin()
    {
        // Unpin, then delete the feed item locally.
        var tileInfo = new TileInfo(this.FormatTileIdUrl());
        this.IsFeedItemPinned = !await this.secondaryPinner.Unpin(tileInfo);

        if (!this.IsFeedItemPinned)
        {
            await this.RemovePinnedFeedItem();
        }
    }

    private string FormatTileIdUrl()
    {
        var queryString = string.Format("parameter={0}", FeedItem.Id);
        return string.Format(Constants.SecondaryUriFormat, queryString);
    }
}

```



GdPicture.NET | from \$3,919.47



All-in-one AnyCPU document-imaging and PDF toolkit for .NET and ActiveX.

- Document viewing, processing, printing, scanning, OMR, OCR, Barcode Recognition
- Annotate image and PDF within your Windows & Web applications
- Read, write and convert vector & raster images in more than 90 formats, including PDF
- Includes sample code for: .NET, VB6, Delphi, VC++, C++ Builder, VFP, HTML, Access...
- 100% royalty-free and world leading Imaging SDK



ComponentOne Studio Enterprise 2013 v2 | from \$1,315.60



.NET Tools for the Professional Developer: Windows, HTML5/Web, and XAML.

- Hundreds of UI controls for all .NET platforms including grids, charts, reports and schedulers
- New RadialMenu, RichTextBox and OrgChart for Windows Store Apps
- Ready -to-use Mobile Project Templates and custom Mobile Scaffolding
- Touch support in WPF and Silverlight controls
- Royalty-free deployment and distribution

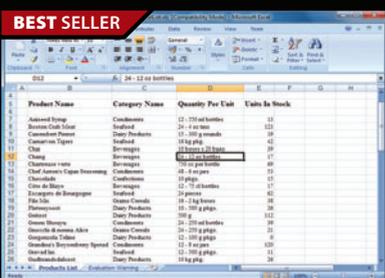


Help & Manual Professional | from \$583.10



Easily create documentation for Windows, the Web and iPad.

- 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 HTML, WebHelp, CHM, PDF, ePUB, RTF, e-book or print
- Styles and Templates give you full design control



Aspose.Total for .NET | from \$2,449.02



Every Aspose .NET component in one package.

- Programmatically manage popular file formats including Word, Excel, PowerPoint and PDF
- Work with charts, diagrams, images, Project plans, emails, barcodes, OCR, and document management in .NET applications
- Common uses also include mail merging, adding barcodes to documents, building dynamic reports on the fly and extracting text from PDF files

both of the apps is to launch the app straight into whatever blog post the secondary tile represents, but allow the user to press the back button and be brought to the main page listing all the blogs, rather than back out of the app itself.

From the Windows 8 perspective, nothing changes from what I talked about in the July 2013 article. **Figure 11** shows the unchanged Windows 8 code for handling when the app is launched, which is a snippet from the App.xaml.cs class file.

It's a little trickier with Windows Phone 8. In this case, the secondary tile takes a uri, rather than just parameters. That means that Window Phone 8 can automatically launch my app into whatever page I want, without going through a centralized launching point first, as Windows 8 does. Because I want to provide a consistent experience across both platforms, I created a centralized launching point of my own for Windows Phone 8, which I called SplashViewModel, shown in **Figure 12**. I set up my project to launch through it whenever the app launches, via a secondary tile or not.

The code for SplashViewModel is pretty straightforward. The one thing I will point out is that I want to make sure to remove this page from the back stack—otherwise it would keep users from ever being able to back out of the app. They would constantly be sent back into the app every time they backed up to this page. This is where the valuable addition to INavigator for Windows Phone 8 comes into play: RemoveBackEntry. After navigating to the MainViewModel, I call RemoveBackEntry to take the splash page out of the back stack. It's now a one-time-use page only for when the app is launched.

Wrapping Up

In this article, I discussed cross-platform development with Windows 8 and Windows Phone 8. I talked about what can be reused between platforms (designs and some code) and what can't

Figure 11 Launching the App from Windows 8

```
protected override async void OnLaunched(LaunchActivatedEventArgs args)
{
    Frame rootFrame = Window.Current.Content as Frame;
    if (rootFrame.Content == null)
    {
        Ioc.Container.Resolve<INavigator>().
            NavigateToViewModel<MainViewModel>();
    }

    if (!string.IsNullOrEmpty(args.Arguments))
    {
        var storage = Ioc.Container.Resolve<IStorage>();
        List<FeedItem> pinnedFeedItems =
            await storage.LoadAsync<List<FeedItem>>(Constants.PinnedFeedItemsKey);
        if (pinnedFeedItems != null)
        {
            int id;
            if (int.TryParse(args.Arguments, out id))
            {
                var pinnedFeedItem = pinnedFeedItems.FirstOrDefault(fi => fi.Id == id);
                if (pinnedFeedItem != null)
                {
                    Ioc.Container.Resolve<INavigator>().
                        NavigateToViewModel<FeedItemViewModel>(pinnedFeedItem);
                }
            }
        }
    }

    Window.Current.Activate();
}
```

Figure 12 SplashViewModel for Windows Phone 8

```
public sealed class SplashViewModel : ViewModelBase<int?>
{
    private readonly IStorage storage;
    private readonly INavigator navigator;

    public SplashViewModel(
        IStorage storage,
        INavigator navigator,
        ISerializer serializer)
        : base(serializer)
    {
        this.storage = storage;
        this.navigator = navigator;
    }

    public override async void LoadState(
        int? navigationParameter, Dictionary<string, object> pageState)
    {
        this.navigator.NavigateToViewModel<MainViewModel>();
        this.navigator.RemoveBackEntry();

        if (navigationParameter.HasValue)
        {
            List<FeedItem> pinnedFeedItems =
                await storage.LoadAsync<List<FeedItem>>(Constants.PinnedFeedItemsKey);
            if (pinnedFeedItems != null)
            {
                var pinnedFeedItem =
                    pinnedFeedItems.FirstOrDefault(fi => fi.Id == navigationParameter.Value);
                if (pinnedFeedItem != null)
                {
                    this.navigator.NavigateToViewModel<FeedItemViewModel>(pinnedFeedItem);
                }
            }
        }
    }
}
```

(XAML). I also talked about some of the challenges faced by developers who work on cross-platform apps, and I presented several solutions to those challenges. My hope is that these solutions can be applied to more than just the specific cases of the navigation, application settings and secondary tiles I talked about. These solutions can help keep your view models testable by providing the ability to abstract out some of the OS interactions and to use interfaces to make these interactions mockable.

In my next article, I'll look more specifically at the actual unit testing of these cross-platform applications, now that they're all set up to be tested. I'll talk more about why I made some of the decisions I made, as they relate to testing, as well as how I actually go about unit testing the apps.

By approaching cross-platform development with the intent of providing similar UXs on both platforms, and by doing a little planning in advance, I can write apps that maximize code reuse and promote unit testing. I can leverage platform-specific features, such as the Windows 8 Charms menu, without sacrificing the experience each platform offers. ■

BRENT EDWARDS is an associate principal consultant for Magenic, a custom application development firm that focuses on the Microsoft stack and mobile application development. He's also a cofounder of the Twin Cities Windows 8 User Group in Minneapolis, Minn. Reach him at brente@magenic.com.

THANKS to the following technical expert for reviewing this article:
Jason Bock (Magenic)

We didn't invent the Internet...

...but our components help you power the apps that bring it to business.



TOOLS • COMPONENTS • ENTERPRISE ADAPTERS

- **E-Business**
AS2, EDI/X12, NAESB, OFTP ...
- **Credit Card Processing**
Authorize.Net, TSYS, FDMS ...
- **Shipping & Tracking**
FedEx, UPS, USPS ...
- **Accounting & Banking**
QuickBooks, OFX ...
- **Internet Business**
Amazon, eBay, PayPal ...
- **Internet Protocols**
FTP, SMTP, IMAP, POP, WebDav ...
- **Secure Connectivity**
SSH, SFTP, SSL, Certificates ...
- **Secure Email**
S/MIME, OpenPGP ...
- **Network Management**
SNMP, MIB, LDAP, Monitoring ...
- **Compression & Encryption**
Zip, Gzip, Jar, AES ...



The Market Leader in Internet Communications, Security, & E-Business Components

Each day, as you click around the Web or use any connected application, chances are that directly or indirectly some bits are flowing through applications that use our components, on a server, on a device, or right on your desktop. It's your code and our code working together to move data, information, and business. We give you the most robust suite of components for adding Internet Communications, Security, and E-Business Connectivity to

any application, on any platform, anywhere, and you do the rest. Since 1994, we have had one goal: to provide the very best connectivity solutions for our professional developer customers. With more than 100,000 developers worldwide using our software and millions of installations in almost every Fortune 500 and Global 2000 company, our business is to connect business, one application at a time.

connectivity
powered by

To learn more please visit our website →

www.nsoftware.com

Using the C++ REST SDK in Windows Store Apps

Sridhar Poduri

In my previous article (msdn.microsoft.com/magazine/dn342869) I introduced the C++ REST SDK and how it can be used in Win32/MFC applications. In this article, I'll discuss how the C++ REST SDK can be integrated within Windows Store apps. One of my original goals in using the C++ REST SDK and the OAuth authentication class was to use as much standard C++ as possible and only interface with platform-specific APIs where necessary. Here's a quick recap of the previous article:

1. Code used in the OAuth authentication class uses standard C++ types and no Windows-specific types.
2. Code used for making Web requests to the Dropbox REST service uses the types from the C++ REST SDK.
3. The only platform-specific code is the function to launch

This article discusses:

- A previous Win32 example app
- Options to integrate with the Windows Runtime
- Using the WebAuthenticationBroker class
- Chaining asynchronous Web requests

Technologies discussed:

C++ REST SDK, OAuth

Code download available at:

archive.msdn.microsoft.com/mag201308CPP

Internet Explorer and complete the application authentication and approval on the Dropbox Apps console portal.

I retained the same goals for my Windows Store app to support authentication and upload a file to Dropbox. I strived to provide the maximum amount of portable C++ code as possible and to interface with the Windows Runtime (WinRT) only where necessary. The example code downloads for both articles can be found at archive.msdn.microsoft.com/mag201308CPP.

Problems with the Win32 Solution

One of the big drawbacks from the previous Win32 application was the need to launch an external application to complete the OAuth authorization process. This meant I had to launch Internet Explorer (you could launch your favorite browser as well), log in to Dropbox using my credentials and then complete the workflow. This is shown in **Figures 1** and **2**.

As you can see, launching an external application and asking users to complete the workflow through the external application takes focus away from my app. As a developer, I also have no standard mechanism through which my app can be notified when the workflow is complete. With my focus on asynchronous programming and with the C++ REST SDK designed to support asynchronous task-based programming, having to launch an external program is clearly a bummer for me. I explored using named pipes, memory-mapped files and so on, but all these approaches require another application to be written to host a Web control instance

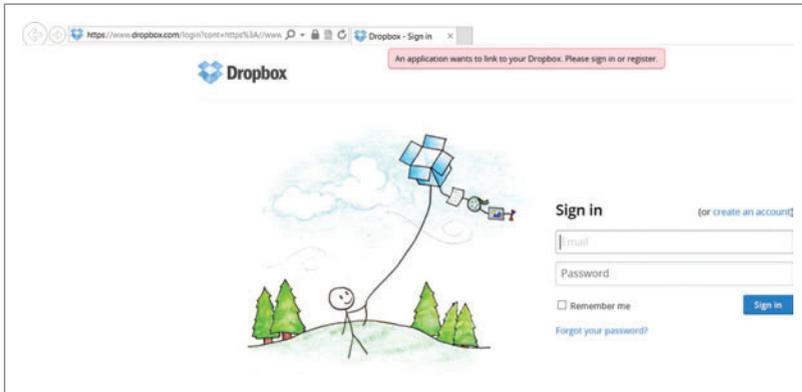


Figure 1 Logging in to Dropbox Using My Credentials Before Authorizing Application Access

and then write the successful value back either via a named pipe, shared memory or a memory-mapped file. I finally settled on using the browser to do the task, as I didn't want to write another application that wrapped the browser control.

One of the big drawbacks from the previous Win 32 application was the need to launch an external application to complete the OAuth authorization process.

Integrating with the Windows Runtime

As I began designing my app to support the Windows Runtime, I considered a few options. I'll briefly list them here and then discuss the chosen approach in more detail:

1. Use protocol activation and let the system launch the right process to handle the protocol by calling the `Windows::System::Launcher::LaunchUriAsync` function. This means that for an HTTPS-based URI, the OS will launch the default browser. This is similar to launching Internet Explorer from the Win32 sample, but with a double whammy: My Windows Store app will be pushed to the background, the default browser will launch full-screen, and—in the worst-case scenario—my app would get suspended by the time the workflow is completed by the user. A big no-no.
2. Integrate the `WebView` control in my app. Using the XAML `WebView` control allows me to embed the entire workflow navigation within the context of my app. Theoretically, I can also be notified if the process has been completed by listening to the `window.external.notify` event fired by the `WebView` control. The reality, though, is that the event is fired only if the Web page fires the notification event. However, in my case, the Dropbox page where the authorization process is completed doesn't fire the event. Bummer!

3. Use the `WebAuthenticationBroker` in my app. As I kept digging around the Windows Runtime, I chanced upon the `WebAuthenticationBroker` class. It looked like it could help me complete the authorization process, and with a few lines of code I was able to get the complete functionality working. Before I dig through code, let me explain the `WebAuthenticationBroker` in some detail.

The WebAuthenticationBroker

In a connected-app world, asking for user credentials via a trusted and secure mechanism is important to gain user consent and approval for an app. No one wants to be the developer whose app leaks user

credentials or is the subject of a stealth attack to hijack user information. The Windows Runtime includes a series of APIs and necessary technologies that allow a developer to seek user credentials in a secure and trustworthy manner. The `WebAuthenticationBroker` is one of the tools that allow for Windows Store apps to use Internet-based authentication and authorization protocols such as OAuth and OpenID. So how does this work in my Dropbox sample app? Here's how:

1. I make the initial asynchronous request to Dropbox, which returns a token and token secret for my app. This initial request is made via the function `oAuthLoginAsync`.
2. Once the `oAuthLoginAsync` function returns, in continuation of the sequence, I construct the URI where the authorization process should begin. In my sample, I've defined the initial URI as a constant string:


```
const std::wstring DropBoxAuthorizeURI =
L"https://www.dropbox.com/1/oauth/authorize?oauth_token=";
```
3. I then build the HTTP request URI by appending the token returned by Dropbox.
4. As an additional step, I construct the callback URI parameter by calling the `WebAuthenticationBroker::GetCurrentApplicationCallbackUri` function. Notice that I didn't use the callback URI parameter in my desktop application because the callback parameter is optional, and I was relying on Internet Explorer to do the authorization task.
5. Now my request string is ready and I can place my request. Instead of using either the C++ REST SDK `http_client` class or `IHttpRequest2` interface to make Web service calls, I call the `WebAuthenticationBroker::AuthenticateAsync` function.

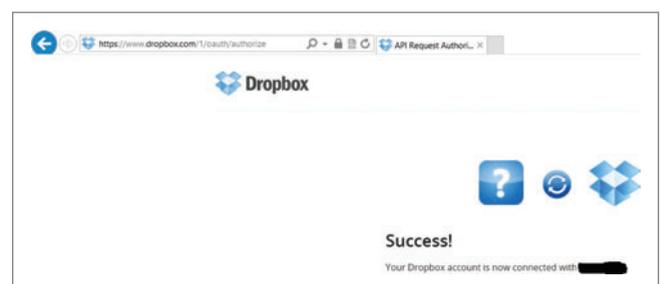


Figure 2 Successful Authorization for My Application on the Dropbox Portal

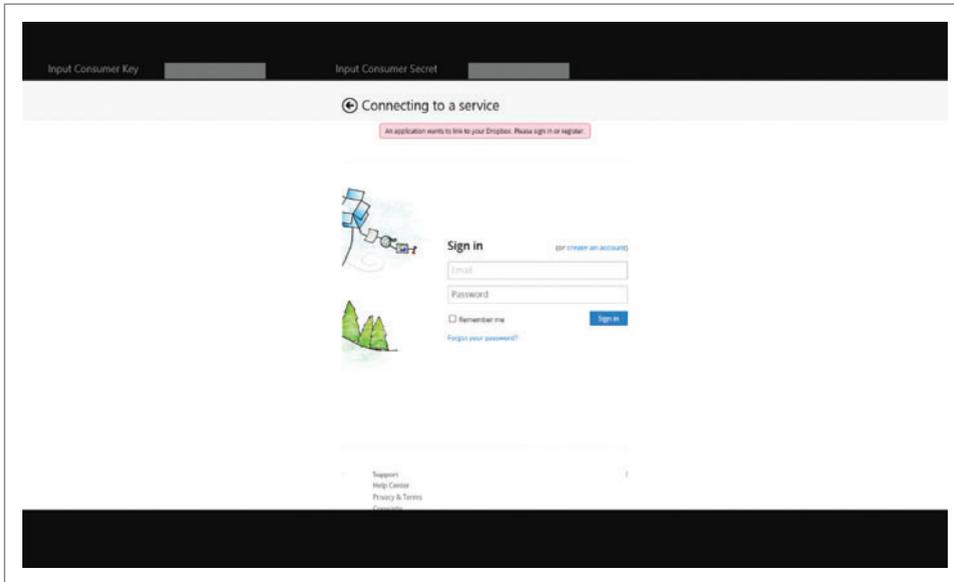


Figure 3 The Sign-in Page of Dropbox as Displayed in the Modal Dialog

6. The `WebAuthenticationBroker::AuthenticateAsync` function accepts two parameters: a `WebAuthenticationOptions` enumeration and a URI. An overloaded instance of the same function accepts the `WebAuthenticationOptions` enumeration and two URIs, one each for the beginning URI where the authentication process starts and the end URI where the authentication process ends.
7. I use the first version of the `AuthenticateAsync` function and pass a `None` value for the `WebAuthenticationOptions` enumeration. For the URI, I pass the URI I've crafted for my Web request.
8. The `WebAuthenticationBroker` sits between my app and the system. At the point where I call `AuthenticateAsync`, it creates a system modal dialog that's a modal to my app.
9. The broker attaches a Web host window to the modal dialog box it has created.

box, clears any persisted cookies created by the Web host from the app container and returns the protocol data back to the app.

In a connected-app world,
asking for user credentials via a
trusted and secure mechanism is
important to gain user consent
and approval for an app.

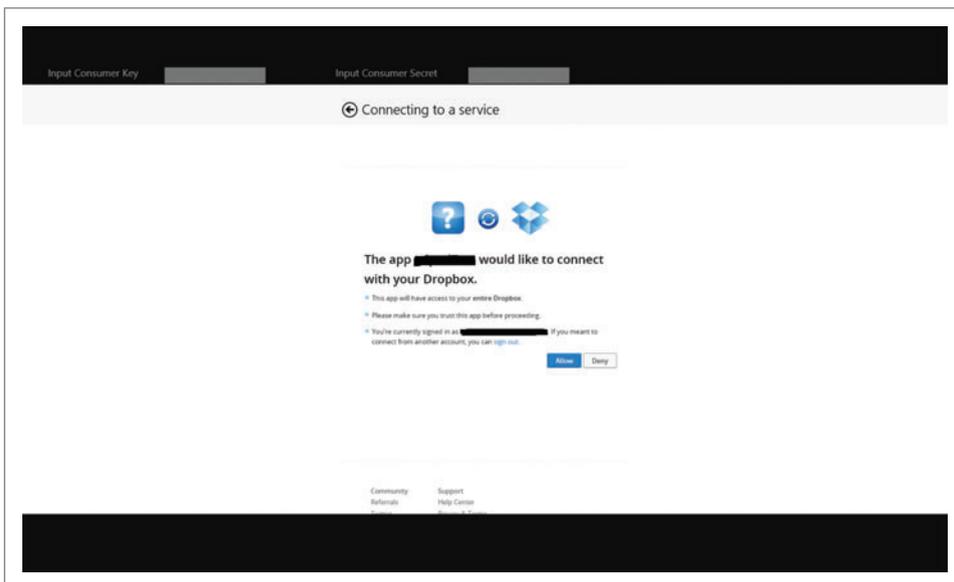


Figure 4 User Consent Being Asked for Application Authorization from Dropbox

10. The broker then selects a dedicated app container process that's separate from the app container in which my app is executing. It also clears any persisted data from my app.
11. The broker then starts the authentication process in this newly selected app container and navigates to the URI as specified in the `AuthenticateAsync` function.
12. As users interact with the Web pages, the broker keeps checking each URL for the callback URI that has been specified.
13. Once a match is found, the Web host ends navigation and sends a signal to the broker. The broker takes down the dialog

Figure 3 illustrates the `WebAuthenticationBroker` modal dialog in my sample Dropbox app after the Web host has navigated to the initial URI. As Dropbox expects users to be signed in before the authorization page can be displayed, the Web host redirects navigation to the Dropbox sign-in page.

Once a user has signed in to Dropbox, the Web host then navigates to the actual authorization URI. This is shown in **Figure 4**. It's clear from both **Figure 3** and **Figure 4** that the dialog is overlaid on top of my app UI. The UI also remains consistent irrespective of the source app calling the `WebAuthenticationBroker::AuthenticateAsync` method. Because the

You've got it, now use it.
Accelerate your development & testing.
You could win* an Aston Martin.



Eligible MSDN subscribers now receive up to \$150** in Windows Azure credits every month for no additional fee. Use virtual machines to accelerate development and test in the cloud. You can activate your benefits in less than five minutes and **no credit card is required*****.



Activate your Windows Azure MSDN Benefits
<http://aka.ms/AzureContest>

*No purchase necessary. Open to eligible Visual Studio Professional, Premium or Ultimate with MSDN subscribers as of June 1, 2013. Ends 11:59 p.m. PT on September 30, 2013. For full official rules including odds, eligibility and prize restrictions see website. Sponsor: Microsoft Corporation. Aston Martin is a trademark owned and licensed by Aston Martin Lagonda Limited.

**Actual credits based on subscription level.

***Customers in multiple countries can now sign up for specified Windows Azure offers without providing a credit card. A credit card may still be required in some countries.

entire experience remains consistent, users can provide credential information without worrying about apps handling credential information and such information leaking accidentally.

One important thing I didn't mention is the need to call the `WebAuthenticationBroker::AuthenticateAsync` function from the UI thread. All C++ REST SDK Web requests are made on a background thread, and I can't display the UI from a background thread. Instead, I use the system dispatcher and call its member function, `RunAsync`, to display the modal UI, as shown in **Figure 5**.

Once a user has signed in to Dropbox, the Web host then navigates to the actual authorization URI.

Once the authorization process is completed, I run the dispatcher again to enable the "Upload File" button on my main UI. This button remains in a disabled state until my users have authenticated and authorized my app to access Dropbox.

Chaining Asynchronous Web Requests

Putting it all together is now easy. In all of the functions that don't interface with the Windows Runtime, I've reused the same code from my desktop application. There are no major code changes except for this: the decision to use the WinRT `StorageFile` object versus a C++ `iostream` in the `UploadFileToDropboxAsync` function.

When writing apps for the Windows Store, there are some limitations that you need to live with. One of the limitations is the need to use WinRT `StorageFile` objects instead of using C++ streams to read and write data from files. When developing a Windows

Figure 5 Using the System Dispatcher to Display the Modal UI

```
auto action = m_dispatcher->RunAsync(
    Windows::UI::Core::CoreDispatcherPriority::Normal,
    ref new Windows::UI::Core::DispatchedHandler([this]()
    {
        auto beginUri = ref new Uri(ref new String(m_authorized.c_str()));

        task<WebAuthenticationResult^> authTask(WebAuthenticationBroker::
            AuthenticateAsync(WebAuthenticationOptions::None, beginUri));

        authTask.then([this](WebAuthenticationResult^ result)
        {
            String^ statusString;
            switch(result->ResponseStatus)
            {
                case WebAuthenticationStatus::Success:
                {
                    auto actionEnable = m_dispatcher->RunAsync(
                        Windows::UI::Core::CoreDispatcherPriority::Normal,
                        ref new Windows::UI::Core::DispatchedHandler([this]()
                        {
                            UploadFileBtn->IsEnabled = true;
                        }));
                }
            }
        }));
    }));
```

Figure 6 The `SignInBtnClicked` Function

```
void MainPage::SignInBtnClicked(Platform::Object^ sender, RoutedEventArgs^ e)
{
    if ((ConsumerKey->Text == nullptr) || (ConsumerSecret->Text == nullptr))
    {
        using namespace Windows::UI::Popups;
        auto msgDlg = ref new MessageDialog(
            "Please check the input for the Consumer Key and/or Consumer Secret tokens");
        msgDlg->ShowAsync();
    }

    m_dispatcher =
        Windows::UI::Core::CoreWindow::GetForCurrentThread()->Dispatcher;

    m_creds = std::make_shared<AppCredentials>();
    m_authenticator = ref new OnlineIdAuthenticator();
    consumerKey = ConsumerKey->Text->Data();
    consumerSecret = ConsumerSecret->Text->Data();

    ConsumerKey->Text = nullptr;
    ConsumerSecret->Text = nullptr;

    OAuthLoginAsync(m_creds).then([this]
    {
        m_authorized = DropBoxAuthorizeURI;
        m_authorized += utility::conversions::to_string_t(this->m_creds->Token());
        m_authorized += L"&oauth_callback=";
        m_authorized += WebAuthenticationBroker::
            GetCurrentApplicationCallbackUri()->AbsoluteUri->Data();

        auto action = m_dispatcher->RunAsync(
            Windows::UI::Core::CoreDispatcherPriority::Normal,
            ref new Windows::UI::Core::DispatchedHandler([this]()
            {
                auto beginUri = ref new Uri(ref new String(m_authorized.c_str()));

                task<WebAuthenticationResult^>authTask(
                    WebAuthenticationBroker::AuthenticateAsync(
                        WebAuthenticationOptions::None, beginUri));

                authTask.then([this](WebAuthenticationResult^ result)
                {
                    String^ statusString;
                    switch(result->ResponseStatus)
                    {
                        case WebAuthenticationStatus::Success:
                        {
                            auto actionEnable = m_dispatcher->RunAsync(
                                Windows::UI::Core::CoreDispatcherPriority::Normal,
                                ref new Windows::UI::Core::DispatchedHandler([this]()
                                {
                                    UploadFileBtn->IsEnabled = true;
                                }));
                        }
                    }
                }));
            }));
    }));
}
```

Store app and using the C++ REST SDK, all operations relating to files expect the developer to pass a `StorageFile` object rather than a C++ stream object. With that one minor change, I'm able to reuse all of my standard C++ code supporting OAuth and Dropbox authorization code in my Windows Store sample app.

Here's my pseudo-code flow (I'll discuss the individual functions after the pseudo-code):

- On clicking the Sign In Button
 - Call `OAuthLoginAsync` function
 - Then call `WebAuthenticationBroker::AuthenticateAsync`
 - Then enable the "Upload File" button on my UI
- On clicking the "Upload File" button
 - Call the `Windows::Storage::Pickers::FileOpenPicker::PickSingleFileAsync` function
 - Then call `OAuthAcquireTokenAsync` function
 - Then call `UploadFileToDropboxAsync` function

WORKFLOW APPLICATIONS | HELP DESK | BUG TRACKING MADE EASY!

Alexsys Team® offers *flexible task management* software for Windows, Web, and Mobile Devices. Track whatever you need to get the job done - anywhere, anytime on practically any device!



Thousands are using Alexsys Team® to create workflow solutions and web apps - without coding! Fortune 500 companies, state, local, DOD, and other federal agencies use Team to manage their tasks. Easily tailor Team to meet your exact requirements - even if they change daily, weekly, or even every minute.

Alexsys Team® Features Include:

- Form and Database customization
- Custom workflows
- Role-based security
- Adaptive Rules Engine
- DOD CAC card support
- Time recording
- Automated Escalations, Notifications, and SOAP Messaging
- Supports MS-SQL, MySQL, and Oracle databases

Our renowned tech support team is here to make you and your team a success. Don't have enough resources? Our professional services staff has helped companies large and small use Alexsys Team® at a fraction of the cost of those big consulting firms.

Find out yourself:

Free Trial and Single User FreePack™ available at Alexcorp.com



FIND OUT MORE!

1-888-880-ALEX (2539)
ALEXCORP.COM

Figure 7 The UploadFileBtnClicked Function

```
void MainPage::UploadFileBtnClicked(
    Platform::Object^ sender, RoutedEventArgs^ e)
{
    using namespace Windows::Storage::Pickers;
    using namespace Windows::Storage;
    auto picker = ref new FileOpenPicker();
    picker->SuggestedStartLocation = PickerLocationId::DocumentsLibrary;
    picker->FileTypeFilter->Append(".txt");

    task<StorageFile^> (picker->PickSingleFileAsync())
        .then([this](StorageFile^ selectedFile)
        {
            m_fileToUpload = selectedFile;
            OAuthAcquireTokenAsync(m_creds).then([this](){
                UploadFileToDropBoxAsync(m_creds);
            });
        });
}
```

In the `SignInBtnClicked` button event handler shown in **Figure 6**, I first perform simple parameter validation to make sure that non-empty string values are passed for the `ConsumerKey` and `ConsumerSecret` parameters that I pass to Dropbox for authentication. Next I obtain an instance of the `Dispatcher` object associated with the `CoreApplication`'s current thread and store it in a member variable of the `MainPage` class. A `Dispatcher` is responsible for processing the window messages and dispatching the events to the app. Next I create an instance of the `OnlineIdAuthenticator` class. The `OnlineIdAuthenticator` class contains helper functions that enable me to pop up an app modal dialog box and complete the secure authorization workflow. This removes the need to launch a browser instance and switch app focus to the browser.

Figure 8 The OAuthAcquireTokenAsync Function

```
task<void> MainPage::OAuthAcquireTokenAsync(
    std::shared_ptr<AppCredentials>& creds)
{
    uri url(DropBoxAccessTokenURI);

    std::shared_ptr<OAuth> oAuthObj = std::make_shared<OAuth>();

    auto signatureParams =
        oAuthObj->CreateOAuthSignedParameters(url.to_string(),
        L"GET",
        NULL,
        consumerKey,
        consumerSecret,
        creds->Token(),
        creds->TokenSecret()
        );

    std::wstring sb = oAuthObj->OAuthBuildSignedHeaders(url);

    http_client client(sb);

    // Make the request and asynchronously process the response.
    return client.request(methods::GET).then([&creds](http_response response)
    {
        if(response.status_code() != status_codes::OK)
        {
            auto stream = response.body();
            container_buffer<std::string> inStringBuffer;
            return stream.read_to_end(inStringBuffer)
                .then([inStringBuffer](pplx::task<size_t> previousTask)
                {
                    UNREFERENCED_PARAMETER(previousTask);
                    const std::string &text = inStringBuffer.collection();

                    // Convert the response text to a wide-character string.
                    std::wstring_convert<std::codecvt_utf8_utf16<wchar_t>,
                    wchar_t> utf16conv;
                    std::wstringstream ss;
                    ss << utf16conv.from_bytes(text.c_str()) << std::endl;
                    OutputDebugString(ss.str().data());
                    // Handle error cases.
                    return pplx::task_from_result();
                });

            // Perform actions here reading from the response stream.
            istream bodyStream = response.body();
            container_buffer<std::string> inStringBuffer;
            return bodyStream.read_to_end(inStringBuffer)
                .then([inStringBuffer, &creds](pplx::task<size_t> previousTask)
                {
                    UNREFERENCED_PARAMETER(previousTask);
                    const std::string &text = inStringBuffer.collection();

                    // Convert the response text to a wide-character string.
                    std::wstring_convert<std::codecvt_utf8_utf16<wchar_t>,
                    wchar_t> utf16conv;
                    std::wstringstream ss;
                    std::vector<std::wstring> parts;
                    ss << utf16conv.from_bytes(text.c_str()) << std::endl;
                    Split(ss.str(), parts, '&', false);
                    unsigned pos = parts[1].find('=');
                    std::wstring token = parts[1].substr(pos + 1, 16);

                    pos = parts[0].find('=');
                    std::wstring tokenSecret = parts[0].substr(pos + 1);
                    creds->SetToken(token);
                    creds->SetTokenSecret(tokenSecret);
                });
        });
    });
}
```

I then call the `OAuthLoginAsync` function, which performs the login operation to Dropbox. Once the async function returns, I use the `Dispatcher` object's `RunAsync` function to marshal the call back to the UI thread from my async task's background thread. The `RunAsync` function takes two parameters: a priority value and a `DispatchedHandler` instance. I set the priority value to `Normal` and pass a lambda function to the `DispatchedHandler` instance. Within the lambda body, I call the static function `AuthenticateAsync` of the `WebAuthenticationBroker` class, which then displays the app modal dialog and helps complete the secure authorization.

When writing apps for the Windows Store, there are some limitations that you need to live with.

Once the workflow is completed, the dialog is taken down and the function returns either a successful completion or any error conditions encountered. In my case, I just handle the `WebAuthenticationStatus::Success` return type and once again use the dispatcher object to enable the `UploadFile` Button on the app UI. Because all the functions I'm calling are async in nature, I need to use the dispatcher object to marshal calls to the UI thread if I want to access any UI elements.

Figure 9 The UploadFileToDropBoxAsync Function

```
task<void> MainPage::UploadFileToDropBoxAsync(
    std::shared_ptr<AppCredentials>& creds)
{
    using concurrency::streams::file_stream;
    using concurrency::streams::basic_istream;

    uri url(DropBoxFileUploadURI);
    std::shared_ptr<OAuth> oAuthObj = std::make_shared<OAuth>();

    auto signatureParams =
        oAuthObj->CreateOAuthSignedParameters(url.to_string(),
        L"PUT",
        NULL,
        consumerKey,
        consumerSecret,
        creds->Token(),
        creds->TokenSecret()
    );

    std::wstring sb = oAuthObj->OAuthBuildSignedHeaders(url);
    return file_stream<unsigned char>::open_istream(this->m_fileToUpload)
        .then([this, sb, url](pplx::task<basic_istream<unsigned char>> previousTask)
        {
            try
            {
                auto fileStream = previousTask.get();
                // Get the content length, used to set the Content-Length property.
                fileStream.seek(0, std::ios::end);
                auto length = static_cast<size_t>(fileStream.tell());
                fileStream.seek(0, 0);

                // Make HTTP request with the file stream as the body.

                http_request req;
                http_client client(sb);
                req.set_body(fileStream, length);
                req.set_method(methods::PUT);
                return client.request(req)
                    .then([this, fileStream](pplx::task<http_response> previousTask)
                    {
                        fileStream.close();
```

Figure 7 shows the UploadFileBtnClicked event handler. There's not much code in the handler itself. I make a call to the FileOpenPicker::PickSingleFileAsync function that allows for selecting a single text file through the picker interface. I then call the OAuthAcquireTokenAsync function as shown in Figure 8 and, upon successful completion, make a call to the UploadFileToDropBoxAsync function shown in Figure 9.

The OAuthAcquireTokenAsync function performs the action to acquire the actual token associated with the Dropbox account. I first build the required access string and the HTTP request headers and call the Dropbox service for performing the credential validation. This HTTP request is of type GET and the response is returned as a stream of characters. I parse the stream and split the character stream to obtain the actual token and token secret values. These are then stored in the AppCredentials class instance.

Once I've successfully obtained the actual token and token secret values from Dropbox, it's time to put them to use by uploading a file to Dropbox. As is the norm with any Dropbox Web endpoint access, I first build the parameter string and HTTP headers. I then call the Dropbox service endpoint associated with uploading files. This HTTP request is of type PUT because I'm attempting to place content on the service. Before I place content, I also need to let Dropbox know about the size of the content. This is specified by setting the content_length property of the HTTP_request::set_body method to the size of the file being uploaded. Once the PUT

```
    std::wstringstream ss;
    try
    {
        auto response = previousTask.get();
        auto body = response.body();
        // Log response success code.
        ss << L"Server returned status code "
        << response.status_code() << L"."
        << std::endl;
        OutputDebugString(ss.str().data());
        if (response.status_code() == web::http::status_codes::OK)
        {
            auto action = m_dispatcher->RunAsync(
                Windows::UI::Core::CoreDispatcherPriority::Normal,
                ref new Windows::UI::Core::DispatchedHandler([this]()
                {
                    using namespace Windows::UI::Popups;
                    auto msgDlg = ref new MessageDialog(
                        "File uploaded successfully to Dropbox");
                    msgDlg->ShowAsync();
                }));
        }
        catch (const http_exception& e)
        {
            ss << e.what() << std::endl;
            OutputDebugString(ss.str().data());
        }
    });
    catch (const std::system_error& e)
    {
        // Log any errors here.
        // Return an empty task.
        std::wstringstream ss;
        ss << e.what() << std::endl;
        OutputDebugString(ss.str().data());
        return pplx::task_from_result();
    }
});
}
```

method returns successfully, I use the dispatcher object to display a success message to the user.

Linux Is Next

Integrating the C++ REST SDK in Windows 8 applications (both Windows Store and desktop) is simple and straightforward. Add the benefits of writing code that can be shared between both the platforms, of using modern C++ programming idioms, and the fact the code is portable across both Windows and non-Windows apps, and you have a winner. You can stop worrying about platform-specific intricacies as they're related to network APIs and instead use your time to think about features that your application should support. In this simple example, I've used the C++ REST SDK to authenticate a user with Dropbox and then upload a file to the Dropbox cloud. For more information about the Dropbox REST API, you can refer to the documentation at bit.ly/100dTD0. In a future article, I'll show you how to accomplish this same task from a Linux client. ■

SRIDHAR PODURI is a program manager on the Windows team at Microsoft. A C++ aficionado and author of the book, "Modern C++ and Windows Store Apps" (Sridhar Poduri, 2013), he blogs regularly about C++ and the Windows Runtime at sridharpoduri.com.

THANKS to the following Microsoft technical experts for reviewing this article: Niklas Gustafsson, Sana Mithani and Oggy Sobajic

Getting Started with Debugging Windows Store Apps

Bruno Terkaly and Robert Evans

In the 1940s, when one of Admiral Grace Hopper's programs failed to work correctly, she discovered a moth stuck between the relays on the Harvard Mark II—and the term “bug” was born. Today, of course, debugging has nothing to do with insects. It's all about the process of finding and removing defects in software.

Despite the fact that today's programming languages and tools go a long way toward helping developers write robust code, there's no protection from logic errors. Modern compilers can only

perform rudimentary checks against programming language syntax and proper use of the type system. The compiler does little to help you determine the actual runtime behavior of your application. That's where debugging skills are essential. This article will focus on debugging Windows Store apps and explore some of the techniques used in the field.

We'll be using Visual Studio 2013 Preview, though these techniques will generally work with earlier versions of the product. Before getting started, however, you'll need to prepare your environment for more advanced debugging than Visual Studio alone provides. We recommend downloading and installing the following resources:

- Windows 8.1 Preview: bit.ly/12nmBEg
- Visual Studio 2013 Preview: bit.ly/1aMpbeM
- Hands-on labs for Windows 8: bit.ly/QHh0JR
- Background task sample (Windows 8.1): bit.ly/1ZpfqN
- Windows SDK for Windows 8.1 Preview: bit.ly/1cM9pyQ
- JustDecompile: bit.ly/qXxKJs
- Process Explorer: bit.ly/fzWyfq

Process Lifecycle Management

With typical desktop applications, the user launches an app and it runs until he terminates it. The application continues to run even when it's not in the forefront. Windows Store apps are a bit different. When an app isn't in the forefront, a Windows Service—the Runtime Broker—suspends all threads in the app, and wakes them back up only when the app is in the forefront. This novel feature helps

GET HELP BUILDING YOUR WINDOWS STORE APP!

Receive the tools, help and support you need to develop your Windows Store apps.

bit.ly/XLjOrx

This article refers to product versions not yet released. All information is subject to change.

This article discusses:

- Process Lifetime Management
- Background tasks and activations
- Using Process Explorer to explore a running app
- Exploring code with JustDecompile

Technologies discussed:

Windows 8.1 Preview, Visual Studio 2013 Preview, Windows SDK for Windows 8.1 Preview

Visual Studio **LIVE!**
EXPERT SOLUTIONS FOR .NET DEVELOPERS

ORLANDO



Access 3 of the best attended sessions on the hottest subjects from Visual Studio Live! Chicago here FREE:



WE'VE GOT YOUR TICKET TO FREE SESSIONS

PREVIEW VISUAL STUDIO LIVE! EVENT CONTENT AT NO CHARGE!

Flip over for more details



ACCESS 3 SESSIONS ON THESE HOT TOPICS:

Beyond Hello World: A Practical Introduction to Node.js

Speaker: *Rick Garibay*

- Node.js programming fundamentals on Windows
- Design considerations for building a URL shortening service
- Building Node.js services that embrace REST/Hypermedia principles

SQL Server Data Tools

Speaker: *Leonard Lobel*

- The declarative model-based approach used in the new generation of SQL Server tools for developers
- Understand the various services that power the new tools
- See live demonstrations of how to design, test, and deploy all from inside Visual Studio

Tips for Building Multi-Touch Enabled Web Sites

Speaker: *Ben Hoelting*

- The new multi-touch enabled capabilities of IE 10
- The new HTML5\CSS3 capabilities of IE 10
- Tips and Tricks for using these capabilities

SCAN THE
QR CODE
TO GET STARTED
TODAY!

Brought to you by
Visual Studio Live!
Orlando



vslive.com/orlando

EVENT SPONSOR

Microsoft

SUPPORTED BY



Visual Studio

msdn
magazine

Visual Studio
MAGAZINE

PRODUCED BY

1105 MEDIA

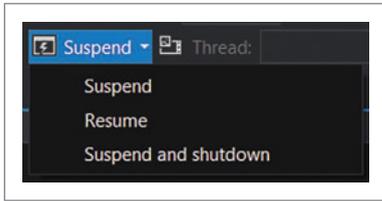


Figure 1 The Debug Location Toolbar

it. If you prefer the older Windows 8 functionality, you can set `Windows.ApplicationModel.ClosePolicy.TerminateOnClose` equal to `true`.

Both Suspend and Resume events are sent to the app, giving you the opportunity to store and retrieve state or take other actions as needed. In addition, if an app applies too much memory pressure or takes too long to start up, the Runtime Broker will terminate the app, killing all threads. Visual Studio automatically disables Process Lifecycle Management (PLM) for apps that are being debugged, whether you're debugging your own app or attaching to an installed app (using Debug | Debug Installed App Package). It's important that PLM is disabled because it would prevent you from being able to properly debug an application. The reason is simple—the Runtime Broker may step in and terminate your application before you have a chance to properly debug it.

However, some of the debuggers outside Visual Studio, such as Windows Debugger (WinDbg) and Microsoft Console Debugger (CDB), require you to manually disable PLM features. To manually disable PLM, you can use the `PLMDebug` tool available in the Windows 8.1 SDK. `PLMDebug` is a command-line tool that allows you to disable PLM for a specific `.appx` package using the `/enableDebug` switch. You can view all installed app packages via the Windows PowerShell command `Get-AppxPackage` or via Process Explorer (as described later in this article) to identify the specific `.appx` package ID.

With Windows 8.1, even when an app is closed by the user, the default action is to place the app into a suspended state and not terminate it.

`PLMDebug` has several handy features. For example, it lets you explicitly send a variety of events to your app, including Suspend, Resume, Terminate, Force-terminate and Clean-terminate. It also allows you to attach a live debugger (which we won't cover in this article).

Visual Studio also lets you trigger Suspend and Resume events, using the Debug Location toolbar, which isn't visible by default. You can add it by selecting View | Toolbars | Debug Location. The debugging options will be enabled once you're actively debugging your app. Figure 1 shows the Debug Location toolbar in Visual Studio 2013 Preview.

Background Tasks

Many scenarios, such as updating a live tile, can be accomplished without using a background task, but sometimes background tasks are necessary and they represent another way in which code in your Windows Store app can be started implicitly due to specific system conditions you define. For example, you might want to add or create thumbnails in the background for one of your applications, but you only want to do it when the device is using AC power. The Maintenance Trigger allows you to start some tasks based on time intervals and system conditions, and to execute the tasks repeatedly. Although the code for background tasks is defined within your app, in most cases it's executed by a separate process named `BackgroundTaskHost`.

Many scenarios, such as updating a live tile, can be accomplished without using a background task.

A Windows Store app registers its background tasks with the OS-level background-task infrastructure by using the `BackgroundTaskBuilder` class. The background task is created as a class that implements the `IBackgroundTask` interface, and you simply need to implement the `Run` method. A background task must have exactly one trigger and one or more conditions set that describe the exact circumstances in which to launch the background task. The trigger is specified with the `SetTrigger` method of the `BackgroundTaskBuilder` class.

Visual Studio makes finding and debugging registered background tasks easy, because the Debug Location toolbar will show your application's declared background tasks and allow you to start them through the Visual Studio visual interface. This technique works for starting all background tasks, except those that use `ControlChannelTrigger`, `PushNotificationTrigger` or a `SystemTrigger` with the `SmsReceived` trigger type. For more information, see the "Introduction to Background Tasks" white paper at aka.ms/035jqc. And you'll find a good code sample at bit.ly/IZpfqN.

Activations

There are many ways your app can be launched depending on what you've implemented, such as File, Protocol, `PrintTaskSettings` or

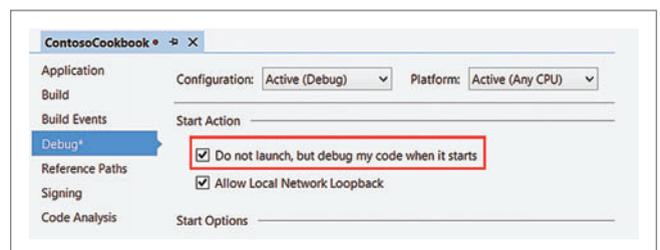


Figure 2 Setting Debugger Properties

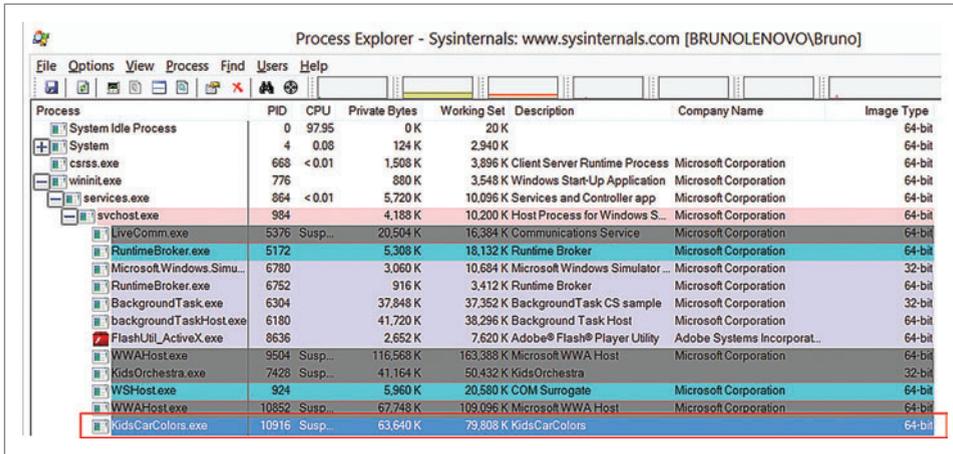


Figure 3 Process Explorer from SysInternals

ShareTarget. For example, the ShareTarget activation type would be active when a user shares something from another app to your app.

There have been some changes in Windows 8.1 regarding activations. Search activation, for example, has been deprecated, but there's still some backward-compatibility support. There's also a new, innovative approach to launching other applications that share the screen with your app. This means you can share the screen with a browser while your app is still active. You'll find more information on this capability at bit.ly/11cKVS3.

To debug your application from any of these types of activations, your first step is to go to your application project properties Start Actions. Start by right-clicking on the ContosoCookbook in the Visual Studio Solution Explorer and choosing Properties. Select the checkbox "Do not launch, but debug my code when it starts," as seen in Figure 2. With this setting you'll be able to debug your application and place breakpoints in your OnLaunched handler that will be triggered when activated by any of the other activation types. This setting tells the application to get ready for debugging but not actually launch when you start a debugger session. The application will simply wait.

Exploring a Running Application

Process Explorer provides some useful features for when you want to get a peek at the inner workings of a running application. To see these features, we'll explore some of the code in an app called Kids Car Colors, which you can download from bit.ly/YnmAxT. Go to the Start screen and start the Kids Car Colors app, then start Process Explorer (assuming you've previously set it up).

If you expand `svchost.exe`, you'll see the Runtime Broker and any executing Windows Store apps, as shown in Figure 3. By right-clicking on `KidsCarColors.exe`, you can view the properties, such as the root folder installation.

The Properties window of an application in Process Explorer provides a wealth of information. For example, as Figure 4 shows, the Image tab lets

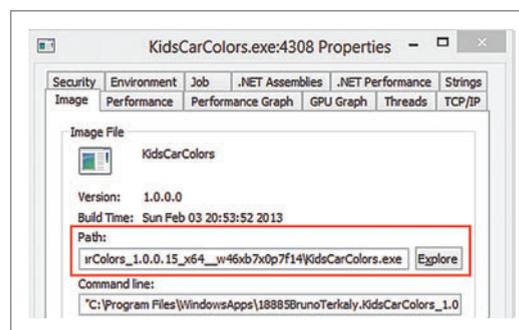


Figure 4 Properties of the Kids Car Colors App

you view the location of a given installed app. Both built-in apps and applications downloaded from the Windows Store always install in the `[Root Folder]\Program Files\Windows Apps` folder.

You'll also find the .NET Performance tab useful; it provides basic performance counter information, including such things as heap sizes, the number of Gen 0 Collections and the percentage of Time spent in garbage collection (GC). Looking at these performance counters allows you to evaluate the effectiveness of the GC, which is the mechanism that frees memory when objects are no longer referenced. It may be necessary to modify an application's code to more aggressively free memory when it's not needed. You can learn more about .NET performance counters at msdn.microsoft.com/library/w8f5kw2e.

To debug your application from any of these types of activations, your first step is to go to your application project properties Start Actions.

You may need to change the ownership and permissions of the `C:\Program Files\Windows Apps` folder so that you can view the applications inside of it.

Inside `C:\Program Files\Windows Apps`, parallel folders exist for each of the installed applications. Each of these application folders has a wealth of information, as described in the next section.

Exploring the Code

As noted, you can go to the install directory for individual applications and find a great deal of information about the application, such as all the MP3 files (if any), images, XAML files and other resources. However, you won't be able to examine the XAML files for Windows 8.1 applications, as they've been compiled into a binary. But there's a variety of tools that lets you decompile the Windows Store app executables written in C# and Visual Basic .NET.

We'll decompile `KidsCarColors.exe` using the JustDecompile tool from Telerik. Decompiling is the process of taking an application binary and

Creating a report is as easy as writing a letter



Reuse MS Word documents as your reporting templates



Create encrypted and print-ready Adobe PDF and PDF/A



Royalty-free WYSIWYG template designer



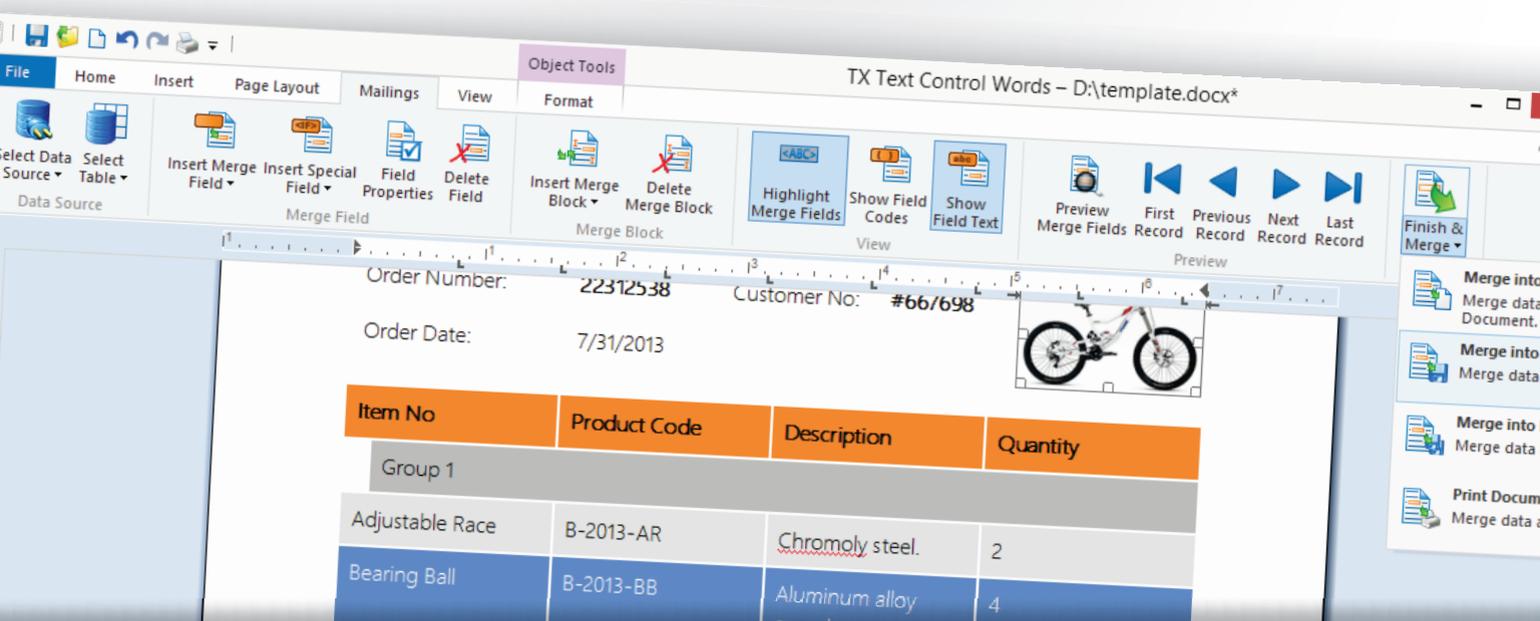
Powerful and dynamic 2D/3D charting support



Easy database connections and master-detail nested blocks



1D/2D barcode support including QRCode, IntelligentMail, EAN



www.textcontrol.com/reporting



txtextcontrol

US: +1 855-533-TEXT
 EU: +49 421 427 067-10



Visual Studio

Microsoft

Partner

Reporting

Rich Text Editing

Spell Checking

Barcodes

PDF Reflow

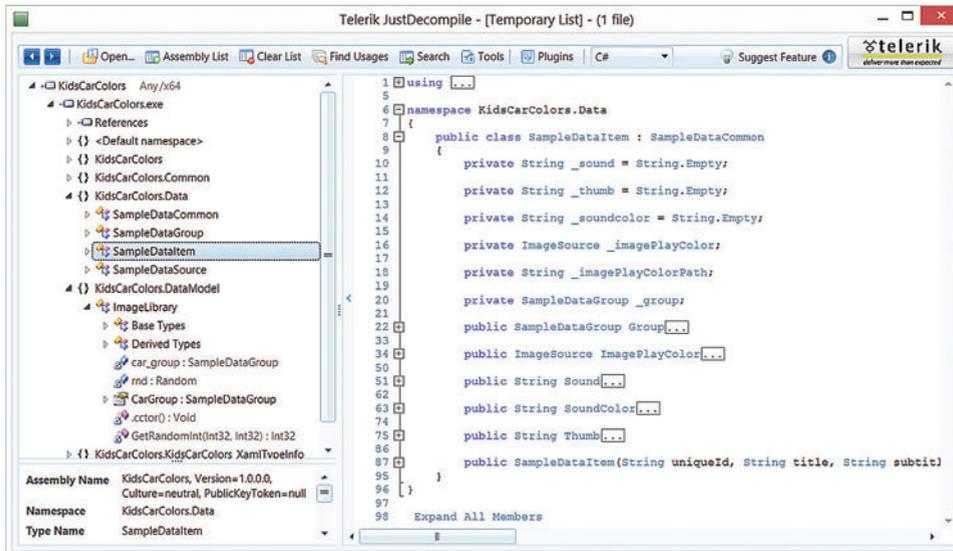


Figure 5 Decompiling KidsCarColors.exe

generating readable source code. To do so, simply navigate to the app's install folder as explained previously, right-click on KidsCarColors.exe and choose Open with JustDecompile. As shown in Figure 5, you can now see low-level details of the code revealed by the decompiler. Note that the data model inside the application is easy to decipher and browse. JustDecompile makes it easy to recover lost source code or peer into assemblies to discover the root cause of an external bug. This means you can view the source code of practically any installed Windows Store application.

It's remarkable how much information is available with this tool. In creating Kids Car Colors, it took some effort to get the audio to work well when a kid clicked on the car color. But all of that code is exposed. If you navigate to the ItemDetailPage object in the JustDecompile project explorer, you'll be able to discover exactly how the sound files are played: retrieving the MP3 from local storage, instantiating various MediaElements, registering media callback events, calling SetSource and so on. For anyone wanting to replicate this approach, it's all there, exposed to the world.

JustDecompile makes it easy to recover lost source code or peer into assemblies to discover the root cause of an external bug.

JustDecompile provides a few buttons on its toolbar, including the Assembly List button, which allows you to explore the references the application has set. Just knowing what references a Windows Store application has set tells you something about what functionality the application uses. For example, you can see that Kids Car Colors leverages the Advertising SDK. That makes sense, because

the app does show advertisements. Theoretically, however, you could set a reference to an assembly that you never use. Imagine that we removed the advertising inside the app but forgot to remove the reference to the Advertising SDK assembly. JustDecompile enables you to see exactly which version is running out in the wild and what the code and references look like.

JustDecompile also includes a feature called Find Usages, which allows you to replicate the Find All References command in Visual Studio. You can simply ctrl-click on any Namespace, Type or Member in your decompiled code to get a list of all corresponding code

references. This can be a tremendous help in learning about how an application works without having any source code.

The Assembly List button allows you to explore the references the application has set.

Wrapping Up

To maximize your effectiveness as a Windows Store app developer, you need to have a good understanding of how to debug applications. This article gives you a brief overview of some tools and techniques you can use to find and fix problems in your code. Moreover, some of the techniques can help you learn how other Windows Store apps work, even if you don't have the source code. ■

BRUNO TERKALY is a developer evangelist for Microsoft. His depth of knowledge comes from years of experience in the field, writing code using a multitude of platforms, languages, frameworks, SDKs, libraries and APIs. He spends time writing code, blogging and giving live presentations on building cloud-based applications, specifically using the Windows Azure platform. You can read his blog at blogs.msdn.com/b/brunoterkalay.

ROBERT EVANS is a premier field engineer and the technical lead for Windows 8: Windows Store App Labs. He's a Microsoft Certified Professional Developer and a Windows 8 Dev Bootcamp Master Instructor. He has presented at TechReady, GeekReady and The Tablet Show, and hosted the Windows 8 Hardware Lab at the Microsoft Build conference in addition to numerous Windows 8 Hackathon events. Prior to becoming a premier field engineer, Evans spent 12 years as a software development engineer at Microsoft working on various products such as Xbox Live, MSN and Mobile Engineering in Microsoft IT. You can read the Premier Field Engineering blog, to which he contributes, at aka.ms/Utq864.

THANKS to the following technical expert for reviewing this article: Christophe Nasarre-Soulier (Microsoft)



Extreme Performance & Linear Scalability

Remove data storage and database performance bottlenecks and scale your applications to extreme transaction processing (XTP). NCache lets you cache data in memory and reduce expensive database trips. It also scales linearly by letting you add inexpensive cache servers at runtime.

Enterprise Distributed Cache

- Extremely fast & linearly scalable with 100% uptime
- Mirrored, Replicated, Partitioned, and Client Cache
- NHibernate & Entity Framework Level-2 Cache

ASP.NET Optimization in Web Farms

- ASP.NET Session State storage
- ASP.NET View State cache
- ASP.NET Output Cache provider
- ASP.NET JavaScript & image merge/minify

Runtime Data Sharing

- Powerful event notifications for pub/sub data sharing

Download a 60-day FREE trial today!



www.alachisoft.com

1-800-253-8195





Visual Studio **LIVE!**

EXPERT SOLUTIONS FOR .NET DEVELOPERS



WE'VE GOT YOUR TICKET TO CODE!

INTENSE TAKE-HOME TRAINING FOR DEVELOPERS,
SOFTWARE ARCHITECTS AND DESIGNERS

Celebrating 20 years of education and training for the developer community, Visual Studio Live! returns to Orlando – and we've got your ticket to Code! Visual Studio Live! Orlando is where developers, software architects and designers will connect for five days of unbiased and cutting-edge education on the Microsoft platform.

YOUR BACKSTAGE PASS TO THE MICROSOFT PLATFORM



WHETHER YOU ARE AN:

- 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 BEFORE
SEPTEMBER 12 AND
SAVE \$400!**

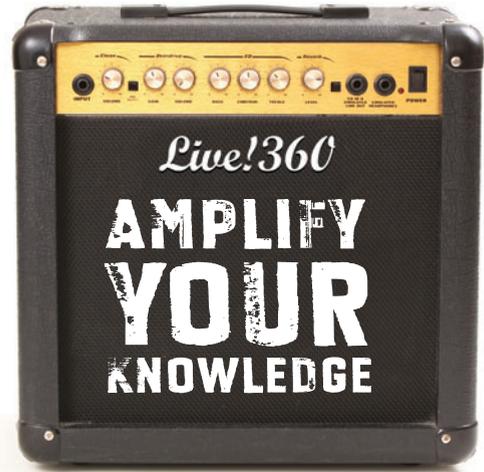
Use Promo Code ORLSEP4



Scan the QR code to register or for more event details.

ORLANDO | **NOVEMBER
18-22, 2013**

Royal Pacific Resort at Universal Orlando



IT EVENTS WITH PERSPECTIVE

Visual Studio Live! Orlando is part of Live! 360, the ultimate IT and Developer line-up. This means you'll have access to four (4) events, 20 tracks, 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!

Visual Studio **LIVE!**
EXPERT SOLUTIONS FOR .NET DEVELOPERS

SQL Server **LIVE!**
TRAINING FOR DBAs AND IT PROS

SharePoint **LIVE!**
TRAINING FOR COLLABORATION

Modern Apps **LIVE!**
MODERN APPS FROM START TO FINISH



TURN THE PAGE FOR MORE EVENT DETAILS

vslive.com/orlando | live360events.com

Check Out the Additional Sessions for Developers at Live!360



SharePoint LIVE!

TRAINING FOR COLLABORATION

SharePoint Live! features 15+ developer sessions, including:

- Workshop: Everything You Need to Know to Build SharePoint 2013 Apps! - *Kirk Evans*
- What's New in SharePoint 2013 Workflow Development? - *Matthew DiFranco*
- "App-etize" Your SharePoint Solutions - Moving Your Solution to the SharePoint 2013 App Model - *Paul Schaefflein*
- Client Side Development - REST or CSOM? - *Mark Rackley*
- TypeScript Development in SharePoint - *Robert Bogue*
- Workshop: Everything You Wanted to Know about Workflow in SharePoint 2013 - *Andrew Connell*

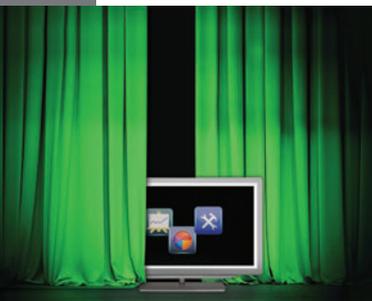


SQL Server LIVE!

TRAINING FOR DBAs AND IT PROS

SQL Server Live! features 20+ developer sessions, including:

- Workshop: SQL Server for Developers - *Leonard Lobel*
- SQL Server Tracing for Developers - *Jason Strate*
- A Window into Your Data: Using SQL Window Functions - *Steve Hughes*
- O, There's My Data, OData for Data Guys - *Steve Hughes*
- Workshop: Big Data and NoSQL for Database and BI Pros - *Andrew Brust*



Modern Apps LIVE!

MODERN APPS FROM START TO FINISH

Modern Apps Live! breaks down the latest techniques in low-cost, high value application development. Sessions include:

- Workshop: Crash Course on Technologies for Modern App Development - *Rockford Lhotka, Nick Landry, & Kevin Ford*
- Modern App Design - *Billy Hollis*
- Building a Modern App for Android in C# with Xamarin - *Nick Landry*
- Building a Modern App in C# and XAML - *Brent Edwards*
- Building a Modern App for iOS - *Lou Miranda*
- Building a Modern App with HTML5 and JavaScript - *Aidan Ryan*

live360events.com



Register at
vslive.com/orlando

Use Promo Code ORLSEP4

Scan the QR code to register or for more event details.

CONNECT WITH
VISUAL STUDIO LIVE!



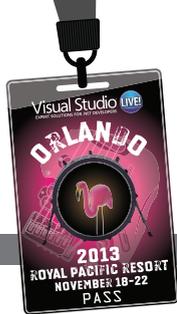
twitter.com/vslive – @VSLive



facebook.com – Search "VSLive"



linkedin.com – Join the "visual studio live" group!



ORLANDO | NOVEMBER 18-22, 2013

Royal Pacific Resort at Universal Orlando

AGENDA AT-A-GLANCE

Windows 8.1 / WinRT		Web and JavaScript Development		Visual Studio / .NET Framework		Azure / Cloud Computing		Mobile Development		Data Management			
Visual Studio Live! Pre-Conference: Sunday, November 17, 2013													
START TIME	END TIME	Pre-Conference Registration - Royal Pacific Resort Conference Center											
4:00 PM	9:00 PM												
6:00 PM	9:00 PM	Dine-A-Round Dinner @ Universal CityWalk											
Visual Studio Live! Pre-Conference Workshops: Monday, November 18, 2013 (Separate entry fee required)													
START TIME	END TIME	Pre-Conference Workshop Registration - Coffee and Morning Pastries											
6:30 AM	8:00 AM												
8:00 AM	5:00 PM	VSM01 - Build an Application in a Day on Windows 8 - <i>Philip Japikse</i>	VSM02 - Rich Data HTML Mobile and Browser Clients with Knockout, JQuery, Breeze, and Web API - <i>Brian Noyes</i>	VSM03 - End-to-End Service Orientation: Designing, Developing, & Implementing Using WCF and the Web API - <i>Miguel Castro</i>									
5:00 PM	7:00 PM	EXPO Preview											
7:00 PM	8:00 PM	Live! 360 Keynote: To Be Announced											
Visual Studio Live! Day 1: Tuesday, November 19, 2013													
START TIME	END TIME	Registration - Coffee and Morning Pastries											
7:00 AM	8:00 AM												
8:00 AM	9:00 AM	Visual Studio Live! Keynote: To Be Announced											
9:00 AM	9:30 AM	Networking Break • Visit the EXPO											
9:30 AM	10:45 AM	VST01 - What's New in Windows 8.1 for Developers - <i>Brian Peek</i>	VST02 - Patterns for JavaScript - <i>John Papa</i>	VST03 - Overview and What's New in Windows Azure - <i>Eric D. Boyd</i>	VST04 - What's New in Visual Studio 2013? - <i>Brian Noyes</i>								
11:00 AM	12:15 PM	VST05 - Controlling Hardware Using Windows 8.1 - <i>Brian Peek</i>	VST06 - Jump-Start for Building Single Page Apps - <i>John Papa</i>	VST07 - IaaS in Windows Azure with Virtual Machines - <i>Eric D. Boyd</i>	VST08 - What's New in the .NET 4.5 BCL - <i>Jason Bock</i>								
12:15 PM	2:00 PM	Lunch • Visit the EXPO											
2:00 PM	3:15 PM	VST09 - A Lap Around Windows Phone 8 Development - <i>David Isbitski</i>	VST10 - WCF & Web API: Can We All Just Get Along?!? - <i>Miguel Castro</i>	VST11 - Solving Security and Compliance Challenges with Hybrid Clouds - <i>Eric D. Boyd</i>	VST12 - How to Be a C# Ninja in 10 Easy Steps - <i>Benjamin Day</i>								
3:15 PM	4:15 PM	Networking Break • Visit the EXPO											
4:15 PM	5:30 PM	VST13 - Developing a Modern Mobile App Strategy - <i>Todd Anglin</i>	VST14 - Building Rich Data HTML Client Apps with Breeze.js - <i>Brian Noyes</i>	VST15 - Cloud Connected Apps with Azure Mobile Services - <i>David Isbitski</i>	VST16 - Software Testing with Visual Studio 2013 and Team Foundation Server 2013 - <i>Benjamin Day</i>								
5:30 PM	7:30 PM	Exhibitor Reception											
Visual Studio Live! Day 2: Wednesday, November 20, 2013													
START TIME	END TIME	Registration - Coffee and Morning Pastries											
7:00 AM	8:00 AM												
8:00 AM	9:00 AM	Live! 360 Keynote: To Be Announced											
9:15 AM	10:30 AM	VSW01 - What's New in WPF 4.5? - <i>Walt Ritscher</i>	VSW02 - Slice Development Time with ASP.NET MVC, Visual Studio, and Razor - <i>Philip Japikse</i>	VSW03 - Build the Next YouTube: Windows Azure Media Services - <i>Sasha Goldshtein</i>	VSW04 - NoSQL for the SQL Guy - <i>Ted Neward</i>								
10:30 AM	11:00 AM	Networking Break • Visit the EXPO											
11:00 AM	12:15 PM	VSW05 - Learning UX Design Principles Through Windows 8 Apps - <i>Billy Hollis</i>	VSW06 - Doing More with LESS for CSS - <i>Todd Anglin</i>	VSW07 - JavaScript, Meet Cloud: Node.js on Windows Azure - <i>Sasha Goldshtein</i>	VSW08 - Gaining the Knowledge of the Open Data Protocol (ODATA) - <i>Christopher Woodruff</i>								
12:15 PM	1:45 PM	Round Table Lunch • Visit the EXPO											
1:45 PM	3:00 PM	VSW09 - Building Windows Store Business Apps with Prism - <i>Brian Noyes</i>	VSW10 - Building Mobile Cross-Platform Apps with HTML5 & PhoneGap - <i>Nick Landry</i>	VSW11 - Applied Windows Azure - <i>Vishwas Lele</i>	VSW12 - Seeking Life Beyond Relational: RavenDB - <i>Sasha Goldshtein</i>								
3:00 PM	4:00 PM	Networking Break • Visit the EXPO • Expo Raffle @ 3:30 p.m.											
4:00 PM	5:15 PM	VSW13 - Migrating from WPF to WinRT - <i>Rockford Lhotka</i>	VSW14 - Connecting to Data from Windows Phone 8 - <i>Christopher Woodruff</i>	VSW15 - Windows Azure in the Enterprise: Hybrid Scenarios - <i>Vishwas Lele</i>	VSW16 - Session To Be Announced								
8:00 PM	10:00 PM	Live! 360 Evening Event											
Visual Studio Live! Day 3: Thursday, November 21, 2013													
START TIME	END TIME	Registration - Coffee and Morning Pastries											
7:00 AM	8:00 AM												
8:00 AM	9:15 AM	VSH01 - Windows 8 HTML/JS Apps for the ASP.NET Developer - <i>Adam Tuliper</i>	VSH02 - Create Data Driven Web Sites with WebMatrix 3 and ASP.NET Web Pages - <i>Rachel Appel</i>	VSH03 - Making the Most of the TFS Service - <i>Brian Randell</i>	VSH04 - iOS Development Survival Guide for the .NET Guy - <i>Nick Landry</i>								
9:30 AM	10:45 AM	VSH05 - Interaction and Navigation Patterns in Windows 8 Apps - <i>Billy Hollis</i>	VSH06 - Building Real-Time, Multi-User Interactive Web and Mobile Applications Using SignalR - <i>Marcel de Vries</i>	VSH07 - Continuous Integration Builds and TFS - <i>Brian Randell</i>	VSH08 - iOS Development Survival Guide for the .NET Guy, Part 2 - <i>Nick Landry</i>								
11:00 AM	12:15 PM	VSH09 - Enhancing the Windows 8 Start Screen with Tiles, Toast and Notifications - <i>Walt Ritscher</i>	VSH10 - Build Data-Driven Mobile Web Apps with ASP.NET MVC and jQuery Mobile - <i>Rachel Appel</i>	VSH11 - IntelliTrace, What is it and How Can I Use it to My Benefit? - <i>Marcel de Vries</i>	VSH12 - Session To Be Announced								
12:15 PM	1:30 PM	Lunch											
1:30 PM	2:45 PM	VSH13 - Adventures in Unit Testing: TDD vs. TED - <i>Ben Hoelting</i>	VSH14 - Maximizing Entity Framework 6 in Your Web Projects - <i>Adam Tuliper</i>	VSH15 - Modern .NET Development Practices and Principles - <i>Jason Bock</i>	VSH16 - Sharing Up to 80% of Code Building Mobile Apps for iOS, Android, WP 8 and Windows 8 - <i>Marcel de Vries</i>								
3:00 PM	4:15 PM	VSH17 - Blend for Visual Studio: Design Your XAML or HTML5/CSS3 UI Faster - <i>Ben Hoelting</i>	VSH18 - Patterns and Tools for Parallel Programming - <i>Marcel de Vries</i>	VSH19 - Static Analysis in .NET - <i>Jason Bock</i>	VSH20 - Talk to Me. Using Speech in Your Windows Phone App - <i>Walt Ritscher</i>								
4:30 PM	5:45 PM	Live! 360 Conference Wrap-up											
Visual Studio Live! Post-Conference Workshops: Friday, November 22, 2013 (Separate entry fee required)													
START TIME	END TIME	Post-Conference Workshop Registration - Coffee and Morning Pastries											
7:00 AM	8:00 AM												
8:00 AM	5:00 PM	VSF01 - Workshop: NoSQL - <i>Ted Neward</i>					VSF02 - Workshop: Visual Studio ALM—To Infinity and Beyond - <i>Brian Randell</i>						

*Speakers and sessions subject to change

Upgrading Windows Phone 7.1 Apps to Windows Phone 8

Michael Crump

While some developers have upgraded their Windows Phone 7.1 projects to take advantage of the new features in Windows Phone 8, others have questions about what it takes to upgrade and what additional tooling and features they should use. In this article, I'll show that going from Windows Phone 7.1 to Windows Phone 8 is easier than you might think.

New Templates

Before looking at the templates, first ask yourself what real-world problems you're going to solve by upgrading. Game developers,

who primarily work with XNA, might want to switch to Direct3D because XNA is no longer actively being developed by Microsoft. You still have the option to create an XNA app, but it won't take advantage of the new features in Windows Phone 8.

Also, native Web developers finally have a chance to get in on the action, as Internet Explorer 10 is preinstalled on the phone. But apart from game and Web development, Windows Phone 8 contains a plethora of new features for XAML/C# and Visual Basic developers.

Here are the new templates included in the Windows Phone 8 SDK:

- **Windows Phone XAML and Direct3D App:** This is a project for creating a Windows Phone managed app with native components. Upon first launch, you'll notice that it comes with two projects: a Windows Phone 8 project and a Windows Runtime (WinRT) Component in C++.
- **Windows Phone HTML5 App:** This is a project for creating a Windows Phone app that primarily uses HTML5 content. This template is often confused with the Windows Library for JavaScript (WinJS) version of Windows Phone, which it is *not*. It's simply using a WebBrowser control to display HTML5 content.
- **Windows Phone Unit Test:** This project contains unit tests that can be used to test Windows Phone apps. This template is added after you install the Visual Studio Update 2 RTM release.

GET HELP BUILDING YOUR WINDOWS PHONE APP!

Receive the tools, help and support you need to develop your Windows Phone 8 apps.

bit.ly/12hQb00

This article discusses:

- New templates, tooling and crucial features in Windows Phone 8
- Performance analysis and testing
- Localization
- Using shared core features of Windows 8

Technologies discussed:

Windows Phone, Windows 8

Upgrading Your Existing Project

Sure, the new project templates help new app development, but what about an existing XAML-based app already built with Windows Phone 7.1? The good news is that you can upgrade such apps to Windows Phone 8 by right-clicking the Windows Phone 7.1 project in Visual Studio 2012 and selecting Upgrade to Windows Phone 8. You'll receive a prompt that cautions you that this upgrade can't be undone and doesn't update any referenced projects. You'll want to make sure that your app is backed up before proceeding. You can also upgrade to Windows Phone 8 by selecting Project Properties, clicking on the

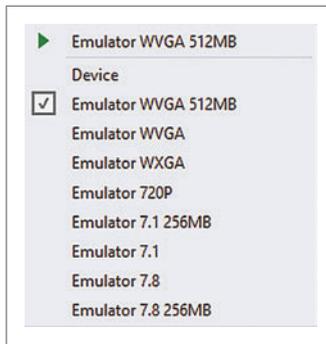


Figure 1 Emulator Options in Windows Phone 8

(found at bit.ly/10pauq4) to add additional emulators to test how your apps will run on Windows Phone 7.8 devices. Because the Windows Phone XAML and XNA app template targets Windows Phone OS 7.1, you can still test your app on a Windows Phone 8 emulator. You can see a list of all the old and new emulators available in **Figure 1**.

With the various emulators available, you no longer have to be dependent on having physical hardware to see your app running on the numerous Windows Phone 7 and 8 devices out there. The new Windows Phone 8 emulators are real virtual machines (VMs) and are one of the best improvements made to the SDK.

Note: You'll need Hyper-V, which is only in Windows 8 Pro or Enterprise, for the new emulators. For more details, see the Windows Phone Dev Center page, "System requirements for Windows Phone Emulator," at bit.ly/QWhAA2.

Also, please keep in mind that with the powerful processors in modern PCs, you should test your app on a physical device before submitting it to the marketplace to gauge real-world performance.

Now that you've seen how the new templates can benefit different sets of developers, and looked at the new emulator options and how easy it is to upgrade your existing project to Windows Phone 8, it's time to start tackling the other important issues that Windows Phone 7 developers have faced.

Using the Simulation Dashboard, you can test in advance of your app going to the marketplace just how well it will perform under different situations.

Application page, selecting Windows Phone OS 8 from the dropdown list and saving your changes.

Also, if you still have a Windows Phone 7 project lying around, you'll be prompted to upgrade it to Windows Phone 7.1 before you can upgrade it to Windows Phone 8. Again, I recommend you back up your project before proceeding.

After your app has been upgraded to Windows Phone 8, you can use the new tooling and SDK features. Now I'll look at all the new emulator options found in Windows Phone 8.

New Emulator Options

In Windows Phone 7.1, you can deploy to only two emulator types with the target screen size of 480x800 (WVGA). The only difference in the emulator images is the amount of RAM (512MB or 256MB). Windows Phone 8 has added two new screen sizes: 768x1280 (WXGA) and 720x1280 (720p). You also have the option of downloading the Windows Phone SDK Update for Windows Phone 7.8

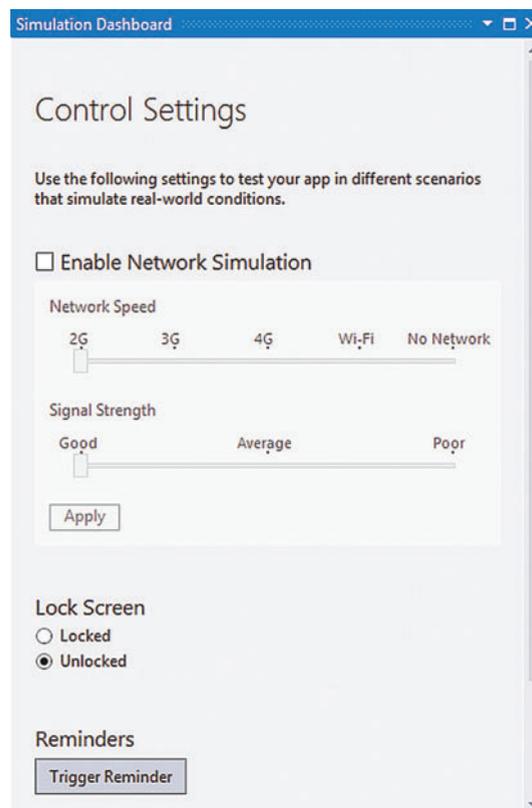


Figure 2 The Simulation Dashboard Included in the Windows Phone 8 SDK

The Simulation Dashboard

When a Windows Phone app is running, a variety of things can interrupt the UX: slow response, having no Internet access, incoming call reminders, the app failing to restore its state after the phone has been locked, and more. In Windows Phone 7.1, you'd probably have to write code that simulated these situations; now you can handle them with the brand-new Simulation Dashboard, as shown in **Figure 2**.

You can access this menu by selecting Tools | Simulation Dashboard from Visual Studio 2012. Using the Simulation Dashboard, you can test in advance of your app going to the marketplace just how well it will perform under different situations.

By enabling Network Simulation and selecting a network speed, you can test various cellular data networks as well as Wi-Fi or scenarios where no networks are available. Particularly interesting are the Signal Strength options, which affect the packet loss rate and network latency. With these options at your fingertips, you should

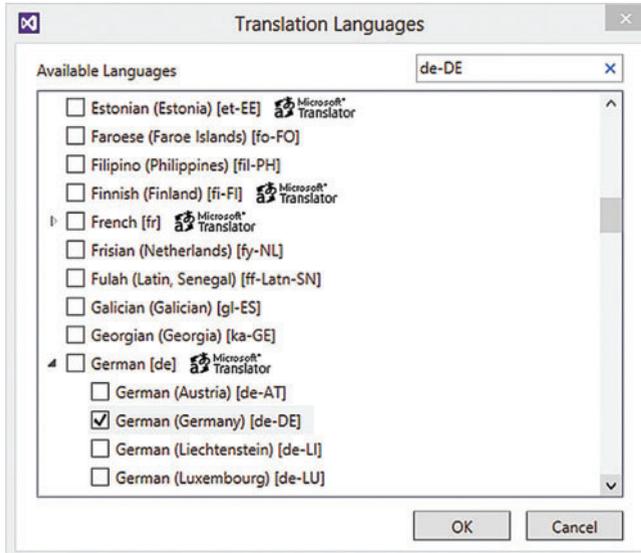


Figure 3 The Translation Languages Dialog Included with the Multilingual App Toolkit

be able to create a Windows Phone 8 app that performs well in a variety of scenarios.

Any app that targets Windows Phone 7.1 or 8 is deactivated once the lock screen has been enabled. It's then activated again once the device has been unlocked. Inside of the Simulation Dashboard, you have the ability to easily lock or unlock the screen to test how your app handles activation or deactivation. You may optionally press the F12 key to show the lock screen.

Finally, you get to use “trigger reminders,” which will simulate an alarm, reminder, phone call, text message or toast notification. Again, you can use these to test how your app handles activation or deactivation.

Windows Phone App Analysis

While the Simulation Dashboard is helpful in providing real-world scenarios that might happen to a user once your app is running on his phone, it doesn't help with the performance of your app. This is where you can make use of Windows Phone app analysis, which can be found at the Debug | Start Windows Phone Application Analysis menu.

This tool provides app monitoring, which helps evaluate the start time and responsiveness as well as profiling. This helps you evaluate

either execution- or memory-related issues in your app. The profiling execution option includes advanced settings that enable you to do things such as visual profiling and code sampling, while the memory options allow you to collect memory allocation stacks and object references. Both of these options will result in a graph displayed in Visual Studio 2012 as well as a time-stamped .sap file added to your project. With the generated graphs, you can drill down into specific start and stop times and see the observation summary that Visual Studio 2012 has generated. The Windows Phone Application Analysis tool is an integral part of your quality assurance process.

Store Test Kit

After you've tested your app under different user scenarios and tested app performance with the help of the Windows Phone Application Analysis kit, you need to test your app to make sure it's certifiable in the Windows Phone Store. This is a vital step, as 30 minutes now can save you several days of lost time if the app fails something that would've been caught by using this kit.

The kit can be easily accessed by right-clicking on your app and selecting Open Store Test Kit. Windows Phone 7.1 also included this functionality, but it was called the Marketplace Test Kit. New and improved tests that target Windows Phone 8 have been added.

Upon first launch, you might see a message with a blue background at the bottom of your screen saying, “Store test cases have been updated. Would you like to install the updated test cases?” You can select Update and download new or modified tests.

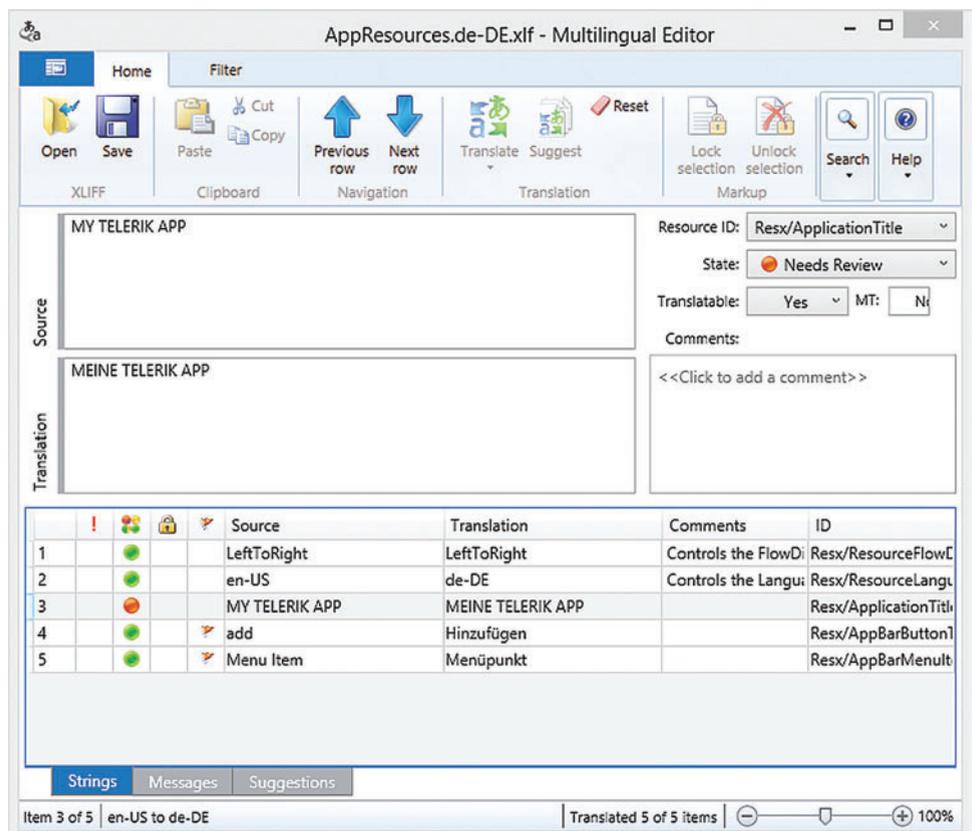


Figure 4 Translating from One Language to Another

This is useful because you always know you're working with the latest available tests.

At the left of your screen are three tabs: Application Details, Automated Tests and Manual Tests. Application Details makes sure the image resources adhere to the guidelines in the Windows Phone Store. This includes the Store Tile as well as the app screenshots for WVGA, WXGA and 720p if your project supports these resolutions. Automated Tests check for XAP package requirements, iconography and screenshots. All you need to do to invoke this feature is click on the Run Test button. The final tab contains manual tests; as of this writing, there are 61 manual tests you can perform. You have to manually indicate whether the test passed or not, but full documentation shows how to do so. Manual tests include those for multiple-device support, app closure, responsiveness and so on.

Localization Made Easy

With Windows Phone 7, there was a missed opportunity by many app developers in localizing their apps. This was often due to the fact that they had little or no help with translating from one language to another. The recent release of the Multilingual App Toolkit and new project templates solved this problem.

The default Windows Phone 8 template guides you through localization with built-in comments in the MainPage.xaml file, and it also structures your app with a helper class and Resources folder. Microsoft added the Multilingual App Toolkit that was originally in Windows 8. Once the Multilingual App Toolkit for Visual Studio 2012 (bit.ly/NgggGU) is installed, it's as simple as selecting Tools | Enable Multilingual App Toolkit. After the toolkit has been enabled for your project, select Project | Add Translation Languages to see the languages available, as shown in **Figure 3**.

With the release of Windows 8 came a shared core.

You can filter on the language you want and then press the OK button. It will automatically add the proper language files to your Resources folder. One file in particular that you'll want to pay attention to is the one with the .xlf extension. This is an industry-standard XML Localization Interchange File Format (XLIFF) file that gives you granular control over any pseudo-translation. Double-clicking on it will bring up the Multilingual Editor that will allow you to translate from one language to another by simply clicking the Translate button. You can see an example of this in **Figure 4**.

In **Figure 4**, you can see that it automatically translated several words for me. Once a translation is complete, you can sign off or pass it on to a human translator for review. In this example, the only words that needed review were "MEINE TELERIK APP" because the word "Telerik" isn't in the translation resource. The human translator would realize that Telerik is spelled the same way in German as it is in English, so it can be left as is. You can save this file to get added support for an additional language.

Figure 5 Writing a File to Isolated Storage in Windows Phone 7.1

```
private void WriteFileToIsolatedStorage(string fileName, string fileContent)
{
    using (IsolatedStorageFile isolatedStorageFile =
        IsolatedStorageFile.GetUserStoreForApplication())
    {
        using (IsolatedStorageFileStream isolatedStorageFileStream =
            isolatedStorageFile.CreateFile(fileName))
        {
            using (StreamWriter streamWriter =
                new StreamWriter(isolatedStorageFileStream))
            {
                streamWriter.Write(fileContent);
            }
        }
    }
}
```

An easy way to test this is to change the Application Title in the MainPage.xaml with the following line:

```
<TextBlock Text="{Binding Path=LocalizedResources.ApplicationTitle,
    Source={StaticResource LocalizedStrings}}"
    Style="{StaticResource PhoneTextNormalStyle}" Margin="12,0"/>
```

Then set the phone language to whatever language you specified. In my example, I selected German, and the app title appeared as "MEINE TELERIK APP".

Taking Advantage of the Shared Core

With the release of Windows 8 came a shared core that Windows Phone 8 developers could use. Some of the more notable improvements in the Microsoft .NET Framework 4.5 are async and await support, as well as an easier way to use Isolated Storage.

In Windows Phone 7.1, you'd typically write the code shown in **Figure 5** to write a file to Isolated Storage.

The code in **Figure 5** uses the System.IO.IsolatedStorage namespace, not found in Windows 8. Instead, both Windows 8 and Windows Phone 8 can make use of Windows.Storage and the async/await pattern to prevent performance bottlenecks and enhance the overall responsiveness of your app. Here's an example of how to write the same exact call in Windows Phone 8, taking advantage of the shared core:

```
public async Task WriteFileToIsolatedStorage(
    string fileName, string fileContent)
{
    IStorageFolder applicationFolder = ApplicationData.Current.LocalFolder;

    IStorageFile storageFile = await applicationFolder.CreateFileAsync(
        fileName, CreationCollisionOption.ReplaceExisting);

    using (Stream stream = await storageFile.OpenStreamForWriteAsync())
    {
        byte[] content = Encoding.UTF8.GetBytes(fileContent);
        await stream.WriteAsync(content, 0, content.Length);
    }
}
```

Another namespace heavily used in Windows 8 is HttpClient. Although the Windows Phone 8 SDK still uses the WebClient class

Figure 6 File Size Comparisons Between the Various Tile Types

Tile Size	Flip and Cycle Size (pixels)	Iconic Size (pixels)
Small	159x159	110x110
Medium	336x336	202x202
Wide	691x336	N/A

by default, Microsoft has provided the `HttpClient` class through NuGet. If you simply search for “Microsoft.Net.Http” and install the NuGet package, you can write code such as the following snippet that will work in Windows 8 as well as Windows Phone 8:

```
private async void Button_Click(object sender, RoutedEventArgs e)
{
    var httpClient = new HttpClient();
    var request = await httpClient.GetAsync(new Uri(
        "http://www.microsoft.com/", UriKind.RelativeOrAbsolute));
    var txt = await request.Content.ReadAsStringAsync();
    // Do something with txt, such as MessageBox.Show(txt)
}
```

Crucial New Features

So far I've discussed a variety of ways to help make your transition to Windows Phone 8 easier. I'll now take a look at several new features that your app can't live without.

You can also add a notification to show an icon and a count—of messages, calls and so on—in the notification area on the Windows Phone 8 device.

New Tile Types Windows Phone 7.1 has one tile type called the Flip Tile and one tile size, 173x173, otherwise known as the Medium Tile type. Windows Phone 8 introduces new tile types and sizes:

- **Flip Tile:** This is identical to Windows Phone 7.1 except for the new tile sizes; it flips from the front to the back.
- **Iconic Tile:** This is based largely on the Windows Phone design principles for a modern look.
- **Cycle Tile:** This lets you cycle through up to nine images.

A comparison of tile sizes can be found in **Figure 6**.

Tiles can be easily configured through the `WMAppManifest.xml` file by selecting “Tile Template” and then adding the proper images. You might also set this through codebehind, and a “Flip Tile template for Windows Phone 8” can be found in the Dev Center at bit.ly/10pavKC.

Lock Screen and Notifications In Windows Phone 7.1, you could see only notifications such as mail, text messages and phone calls. Now your users have the ability to use your app as a lock screen

Figure 7 Setting the Phone's Lock Screen Background Image

```
private async void btnLockScreenImage_Click_1(
    object sender, RoutedEventArgs e)
{
    if (!LockScreenManager.IsProvidedByCurrentApplication)
    {
        await LockScreenManager.RequestAccessAsync();
    }

    if (LockScreenManager.IsProvidedByCurrentApplication)
    {
        Uri imageUri = new Uri(
            "ms-appx:///LockScreen.jpg", UriKind.RelativeOrAbsolute);
        Windows.Phone.System.UserProfile.LockScreen.SetImageUri(imageUri);
    }
}
```

background image provider and include custom notifications similar to those described earlier. Setting the background image can be as easy as adding an image to your folder with the content type and updating the app manifest file to declare your app as a background provider. Right-click on the `WMAppManifest.xml` file; choose “Open With” and select XML (Text) Editor; then add this extension:

```
<Extensions>
  <Extension ExtensionName="LockScreen_Background"
    ConsumerID="{1110FF24-AA15-4A96-8006-2BFF8122084F}" TaskID="_default" />
</Extensions>
```

Next, call the code snippet shown in **Figure 7**.

You'll notice that you begin by first checking to see if the user has access to change the background. If not, you'll present a GUI asking for permission, then create a URI with the path to your image and use the `Windows.Phone.System.UserProfile.LockScreen` namespace to set it.

You can also add a notification to show an icon and a count—of messages, calls and so on—in the notification area on the Windows Phone 8 device. For more information on this, see the article, “Lock screen notifications for Windows Phone 8,” at bit.ly/QhyXyR.

Speech One of the most exciting new features is speech. Several speech components are included in the Windows Phone 8 SDK:

- **Text-to-speech (also known as speech synthesis):** This allows text to be spoken back to the user through the phone speaker, headphones or Bluetooth connection.
- **Speech-to-text (also known as speech recognition):** This allows your users to speak commands to a phone to accomplish tasks.
- **Voice commands:** These allow your users to speak commands outside of your app by holding down the start button and saying “open” or “start,” followed by your app name, to perform certain tasks.

All of this is made possible with the `Speech.Synthesis` and `Speech.Recognition` APIs. A simple implementation of text-to-speech can be accomplished with two lines of code in an event handler:

```
private async void Button_Click(object sender, RoutedEventArgs e)
{
    SpeechSynthesizer synth = new SpeechSynthesizer();
    await synth.SpeakTextAsync("The latest MSDN issue has arrived!");
}
```

Just make sure the `async` and `await` operators have been added to the method.

Make the Most out of Your Move

I've discussed everything from the new tooling and templates to some of the new features included in the Windows Phone 8 SDK. I've shown how easy it is to implement localization and described the added bonus of a shared code base with Windows 8. You should now be equipped with the knowledge to make the most out of your move from Windows Phone 7 to Windows Phone 8. ■

MICHAEL CRUMP is a Microsoft MVP, INETA Champion and an author of several .NET Framework e-books. He works at Telerik with a focus on the XAML control suite. You can reach him on Twitter at twitter.com/mbcrumpp or keep up with his blog by visiting michaelcrump.net.

THANKS to the following technical experts for reviewing this article: Jeff Blankenburg (Microsoft) and Lance McCarthy (Telerik)

SpreadsheetGear

Performance Spreadsheet Components

SpreadsheetGear 2012 Now Available

NEW!

WPF and Silverlight controls, multithreaded recalc, 64 new Excel compatible functions, save to XPS, improved efficiency and performance, Windows 8 support, Windows Server 2012 support, Visual Studio 2012 support and more.

Excel Reporting for ASP.NET, WinForms, WPF and Silverlight



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.

Excel Compatible Windows Forms, WPF and Silverlight Controls



Add powerful Excel compatible viewing, editing, formatting, calculating, filtering, charting, printing and more to your Windows Forms, WPF and Silverlight applications with the easy to use WorkbookView controls.

Excel Dashboards, Calculations, Charting and More



You and your users can design dashboards, reports, charts, and models in Excel or the SpreadsheetGear Workbook Designer rather than hard to learn developer tools and you can easily deploy them with one line of code.

**Free
30 Day
Trial**

Download our fully functional 30-Day evaluation and bring Excel Reporting, Excel compatible charting, Excel compatible calculations and much more to your ASP.NET, Windows Forms, WPF, Silverlight and other Microsoft .NET Framework solutions.

www.SpreadsheetGear.com



SpreadsheetGear

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

Adding FTP Support in Windows Phone 8

Uday Gupta

FTP is one of the most widely used protocols for sharing files. It's used to hand off files to clients, to distribute installation files to consumers, to provide access to otherwise inaccessible parts of the file system for security reasons in enterprise environments, and for a host of other scenarios. With all the emphasis on the cloud, it might seem counterintuitive today to continue to rely on FTP. But if you want to have direct control of your files while providing easy access to them, FTP can still be the way to go.

The addition of socket support in Windows Phone enabled the devices to communicate with a multitude of different services, making users feel more connected and available than ever. However, in Windows Phone (as well as Windows 8), it may be up to the developer to provide his own implementation of a service. One feature that's still missing in Windows Phone is support for FTP—

there are no direct APIs available for leveraging FTP services in a Windows Store app. From an enterprise perspective, this makes it rather difficult for employees to access files over their phones.

This article is about providing support for FTP on Windows Phone 8 by creating an FTP library and a simple FTP client application that will run on a Windows Phone device. FTP is a well-established, well-documented protocol (see RFC 959 at bit.ly/fB5ezA). RFC 959 describes the specification, its functionality and architecture, and its commands with their responses. I'm not going to describe every feature and command of FTP, but I hope to provide a head start to help you create your own FTP implementation. Note that the feature I'm going to discuss is not a Microsoft implementation and may not necessarily be approved.

FTP Channels

FTP consists of two channels, a command channel and a data channel. The command channel is created when a client connects to an FTP server, and it's used to transmit all commands and responses to and from the FTP server. The command channel remains open until the client disconnects, a connection-idle timeout takes place, or an error occurs in the FTP server or command channel.

The data channel is used to transmit data to and from the FTP server. This channel is temporary in nature—once the data transfer completes, the channel is disconnected. For every data transmission, a new data channel is established.

An FTP connection can be either active or passive. In active mode, the FTP client sends the data channel information (the data port number) to which the file transfer will be sent. In passive mode,

This article discusses:

- The FTP protocol
- Creating an FTP library and an FTP client app
- Connecting and authenticating
- Browsing directories
- Supporting passive commands

Technologies discussed:

Windows Phone 8, Windows 8, Visual Studio

Code download available at:

archive.msdn.microsoft.com/mag201309WPFTP

Figure 1 Asynchronous Methods and Associated Events

Method (with Parameters)	Associated Events
Constructor(System.String IPAddress, System.Windows.Threading.Dispatcher UIDispatcher)	No associated event
ConnectAsync	FtpConnected
DisconnectAsync	FtpDisconnected
AuthenticateAsync AuthenticateAsync(System.String Username, System.String Password)	Success - FtpAuthenticationSucceeded Failure - FtpAuthenticationFailed
GetPresentWorkingDirectoryAsync	FtpDirectoryListed
ChangeWorkingDirectoryAsync	Success - FtpDirectoryChangedSucceeded Failure - FtpDirectoryChangedFailed
GetDirectoryListingAsync	FtpDirectoryListed
UploadFileAsync(System.IO.Stream LocalFileStream, System.String RemoteFilename)	Success - FtpFileUploadSucceeded Failure - FtpFileUploadFailed Progress - FtpFileTransferProgressed
DownloadFileAsync(System.IO.Stream LocalFileStream, System.String RemoteFilename)	Success - FtpFileDownloadSucceeded Failure - FtpFileDownloadFailed Progress - FtpFileTransferProgressed

WinPhoneFtp.FtpService is the FTP client library that will be referenced in the WinPhoneFtp.UserExperience UX project in order to use FTP services. The FTP library uses an event-based, async/await pattern. Each operation will be called asynchronously in a background task and, upon completion, an event will be raised to provide notification in the UI. The FTP client library contains various asynchronous methods for each FTP operation (the supported commands) and each asynchronous method is mapped to a certain event that will be raised when the asynchronous method completes. **Figure 1** depicts the mapping of asynchronous methods and events.

the client asks the server to create a data channel on its end and provide the socket address and port information so the client can connect to that data channel and begin the file-transfer operation.

Passive mode is useful when the FTP client doesn't want to manage data ports or data channels on its end. While developing the FTP client for Windows Phone, I'll switch over to passive mode. That is, before beginning a file-transfer operation, I'll ask the FTP server to create and open a data channel and send me its socket endpoint information so that when I connect, data transfer begins. I'll discuss how to use passive mode later in the article.

Getting Started

This article won't cover all of the commands mentioned in RFC 959. I'll focus instead on the basic commands that are required to build a minimal, working FTP client, including those involved in the connect-disconnect procedure, authentication, browsing the file system, and uploading and downloading files.

To get started, I'll create a Visual Studio solution that will contain two projects, a Windows Phone Class Library project for the FTP library and a Windows Phone App project to contain code for the UX and for using the FTP library.

To create the solution, open Visual Studio, create a Windows Phone Class Library project and name it WinPhoneFtp.FtpService.

Now add a Windows Phone App project and name it WinPhoneFtp.UserExperience. This will contain the UI for the test app.

The application's UI consists of two textboxes and two buttons. One textbox and button will be used for accepting the FTP server IP address and connecting to it. The other textbox and button will accept FTP commands from a user and send them to the FTP server.

The TCP Socket Wrapper

Before starting on the FTP client, I created a wrapper named TcpClientSocket for the Windows.Networking.Sockets.StreamSocket class (see bit.ly/15fmqhK). This wrapper will provide certain event notifications based on various socket operations. The notifications I'm interested in are SocketConnected, DataReceived, ErrorOccured and SocketClosed. When the StreamSocket object connects to the remote endpoint, the SocketConnected event will be raised. Similarly, on receiving data over the socket, the DataReceived event will be raised, along with the buffer containing data as event arguments. If an error occurs during socket operation, it's the ErrorOccured event that does the notification. And, finally, when the socket is closed (by either a remote or local host), the SocketClosed event is raised, with the reason for the closing in its event argument. I won't go into detail about how the socket wrapper is implemented, but you can download the source code for this article and see its implementation (archive.msdn.microsoft.com/mag201309WPFFTP).

FTP Connect and Disconnect

When the FTP client first connects to the FTP server, it establishes a command channel with the server over which commands and their responses are shared. Typically, FTP uses port 21, but for security reasons or other factors, a different port number may be configured. Before a user can start sharing



Figure 2 Connecting and Authenticating to the FTP Server



Figure 3 Disconnecting from the FTP Server

Figure 4 Sending the Password with PASS, After the USER Command

```

async void FtpClientSocket_DataReceived(object sender, DataReceivedEventArgs e)
{
    String Response = System.Text.Encoding.UTF8.GetString(
        e.GetData(), 0, e.GetData().Length);
    logger.AddLog(String.Format("FTPServer -> {0}", Response));
    switch (ftpPassiveOperation)
    {
        ...

        case FtpPassiveOperation.None:
            switch (ftpCommand)
            {
                case FtpCommand.Username:
                    if (Response.StartsWith("501"))
                    {
                        IsBusy = false;
                        RaiseFtpAuthenticationFailedEvent();
                        break;
                    }

                    this.ftpCommand = FtpCommand.Password;
                    logger.AddLog(String.Format(
                        "FTPClient -> PASS {0}\r\n", this.Password));
                    await FtpCommandSocket.SendData(
                        String.Format("PASS {0}\r\n", this.Password));
                    break;

                ...
            }
        }
    }
}

```

files, he needs to authenticate to the server, typically with a username and password. If the authentication is successful, the server will respond (in my case): 220 Microsoft FTP Service.

Before connecting to the FTP server, I'll create an object named `FtpClient` based on my FTP client library class, along with event handlers for all the events described earlier:

```

FtpClient ftpClient = null;
async private void btnLogin_Tap(object sender,
    System.Windows.Input.GestureEventArgs e)
{
    ftpClient = new FtpClient(txtIp.Text, this.Dispatcher);
    // Add event-handlers to various events of ftpClient object
    await ftpClient.ConnectAsync();
}

```

Once the object is created, and various FTP client event handlers are added, I'll call the `ConnectAsync` method of the `FtpClient` object. Here's how `ConnectAsync` works:

```

public async Task ConnectAsync()
{
    if (!IsConnected)
    {
        logger.AddLog("FTP Command Channel Initailized");
        await FtpCommandSocket.PrepareSocket();
    }
}

```

`ConnectAsync` will connect to the FTP server as a socket-connect operation. When the FTP client successfully connects to the FTP server, the `FtpConnected` event is raised to notify the client that it's connected to the FTP server:

```

async void ftpClient_FtpConnected(object sender, EventArgs e)
{
    // Handle the FtpConnected event to show some prompt to
    // user or call AuthenticateAsync to authenticate user
    // automatically once connected
}

```

Figure 2 shows what the app will look like when a user has successfully connected to the FTP server.

Note that when you connect to other FTP servers, the text you see may be altogether different.

There are three ways to become disconnected from an FTP server:

- The user sends a QUIT command to the server to deliberately close the connection.
- After being idle for a specified time, the FTP server closes the connection.
- A socket error or some internal error may occur on one or both ends.

To deliberately disconnect via the QUIT command, I issue QUIT in the command's textbox and the Send Command button's tap event handler calls the `DisconnectAsync` method:

```

async private void btnFtp_Tap(object sender, System.Windows.Input.
    GestureEventArgs e)
{
    ...
    if (txtCmd.Text.Equals("QUIT"))
    {
        logger.Logs.Clear();
        await ftpClient.DisconnectAsync();
        return;
    }
    ...
}

```

This is how `DisconnectAsync` sends the command to the FTP server:

```

public async Task DisconnectAsync()
{
    ftpCommand = FtpCommand.Logout;
    await FtpCommandSocket.SendData("QUIT\r\n");
}

```

Once the connection is closed, the `FtpDisconnected` event is raised, so that the client can handle the `Disconnect` event and gracefully release the resources, with the result shown in **Figure 3**:

```

void ftpClient_FtpDisconnected(object sender, FtpDisconnectedEventArgs e)
{
    // Handle FtpDisconnected event to show some prompt
    // or message to user or release
}

```

In any of the scenarios mentioned, the FTP client is disconnected from the FTP server and any ongoing FTP file-transfer operations will also abort.

FTP Authentication

Before starting any FTP file operation, the user needs to authenticate to the FTP server. There are two types of users: anonymous and non-anonymous (those with credentials). When the FTP server can be accessed by anybody, users are authenticated as



Figure 5 Getting the Present Working Directory from the FTP Server



Figure 6 Changing Working Directory on the FTP Server

```

FTPClient -> PASV
FTPServer -> 227 Entering Passive Mode
(192,168,33,238,142,237).
FTP Data Channel IPAddress: 192.168.33.238, Port:
36589
FTP Data Channel connected

```

Figure 7 Passive Command Request and Reply

you need to know which type of authentication is enabled on the server.

Authentication takes place via two FTP commands, USER and PASS. The USER command is used to send the user's identity, that is, the username. The PASS command is used to provide the password. Although it's good practice to provide the user's e-mail address, any text formatted as an e-mail address will work:

```

[FTP Client]: USER anonymous
[FTP Server:] 331 Anonymous access allowed, send identity (e-mail name) as password
[FTP Client]: PASS m@m.com
[FTP Server]: 230 User logged in

```

The AuthenticateAsync method has two overloads. The first overload will authenticate the user with default credentials, which are generally used for anonymous authentication. The second overload takes a username and password for authentication to the FTP server.

To authenticate the user automatically as an anonymous user, I call AuthenticateAsync when the FtpConnected event is received:

```

async void ftpClient_FtpConnected(object sender, EventArgs e)
{
    await (sender as FtpClient).AuthenticateAsync();
}

```

Internally, AuthenticateAsync calls the other overload with default credentials:

```

public async Task AuthenticateAsync()
{
    await AuthenticateAsync("anonymous", m@m.com);
}

```

The AuthenticateAsync overload issues the USER command to the FTP server along with the username received from the parameter:

```

public async Task AuthenticateAsync(String Username, String Password)
{
    ftpCommand = FtpCommand.Username;
    this.Username = Username;
    this.Password = Password;
    logger.AddLog(String.Format("FTPClient -> USER {0}\r\n", Username));
    await FtpCommandSocket.SendData(String.Format("USER {0}\r\n", Username));
}

```

Figure 8 Parsing the Data Channel Endpoint

```

private async Task PrepareDataChannelAsync(String ChannelInfo)
{
    ChannelInfo = ChannelInfo.Remove(0, "227 Entering Passive Mode".Length);

    // Configure the IP Address
    String[] Splits = ChannelInfo.Substring(ChannelInfo.IndexOf("(") + 1,
        ChannelInfo.Length - ChannelInfo.IndexOf(")") - 5).Split(
        new char[] { ',', ' ' },
        StringSplitOptions.RemoveEmptyEntries);
    String Ipaddr = String.Join(".", Splits, 0, 4);

    // Calculate the Data Port
    Int32 port = Convert.ToInt32(Splits[4]);
    port = ((port << 8) | Convert.ToInt32(Splits[5]));
    logger.AddLog(String.Format(
        "FTP Data Channel IPAddress: {0}, Port: {1}", Ipaddr, port));

    // Create data channel here with extracted IP Address and Port number
    logger.AddLog("FTP Data Channel connected");
}

```

anonymous. For them, the username is "anonymous" and the password can be any text, formatted as an e-mail address. For non-anonymous FTP, users have to provide valid credentials. Before implementing any authentication scheme, you

When the response to the USER command is received, the PASS command is issued along with the password received from the parameter (see Figure 4).

If the authentication is successful, the FtpAuthenticationSucceeded event is raised; if not, the FtpAuthenticationFailed event is raised:

```

void ftpClient_FtpAuthenticationFailed(object sender, EventArgs e)
{
    logger.AddLog("Authentication failed");
}

void ftpClient_FtpAuthenticationSucceeded(object sender, EventArgs e)
{
    logger.AddLog("Authentication succeeded");
}

```

Directory Browsing

FTP provides support for directory browsing, but this depends on how the FTP client is written to display directory information. If the FTP client is a GUI, the information may be shown as a directory tree. A console client, in contrast, may just display the directory listing on the screen.

Figure 9 Processing the List Command in the Socket's DataReceived Event

```

async void FtpClientSocket_DataReceived(object sender, DataReceivedEventArgs e)
{
    String Response = System.Text.Encoding.UTF8.GetString(
        e.GetData(), 0, e.GetData().Length);

    ...

    IsBusy = true;
    DataReader dataReader = new DataReader(
        FtpDataChannel.InputStream);
    dataReader.InputStreamOptions = InputStreamOptions.Partial;
    fileListingData = new List<byte>();
    while (!(await dataReader.LoadAsync(1024)).Equals(0))
    {
        fileListingData.AddRange(dataReader.DetachBuffer().ToArray());
    }
    dataReader.Dispose();
    dataReader = null;
    FtpDataChannel.Dispose();
    FtpDataChannel = null;
    String listingData = System.Text.Encoding.UTF8.GetString(
        fileListingData.ToArray(), 0, fileListingData.ToArray().Length);
    String[] listings = listingData.Split(
        new char[] { '\r', '\n' }, StringSplitOptions.RemoveEmptyEntries);
    List<String> Filenames = new List<String>();
    List<String> Directories = new List<String>();
    foreach (String listing in listings)
    {
        if (listing.StartsWith("drwx") || listing.Contains("<DIR>"))
        {
            Directories.Add(listing.Split(new char[] { ' ' }).Last());
        }
        else
        {
            Filenames.Add(listing.Split(new char[] { ' ' }).Last());
        }
    }
    RaiseFtpDirectoryListedEvent(Directories.ToArray(),
        Filenames.ToArray());

    ...

    fileListingData = new List<byte>();
    ftpPassiveOperation = FtpPassiveOperation.ListDirectory;
    logger.AddLog("FTPClient -> LIST\r\n");
    await FtpCommandSocket.SendData("LIST\r\n");

    ...
}

```

Because there's no mechanism provided to let you know which directory you're in, FTP provides a command to show the present working directory (the current directory). Sending the PWD command to the FTP server over the command channel gives you the whole path, from the root to the current directory. When a user is authenticated, he lands in either the root directory or the directory configured for him in the FTP server configuration.

You can issue the PWD command by calling `GetPresentWorkingDirectoryAsync`. To get the current directory, I issue PWD in the command textbox and call `GetPresentWorkingDirectoryAsync` in the Send Command button's tap event:

```
async private void btnFtp_Tap(object sender,
    System.Windows.Input.GestureEventArgs e)
{
    ...
    if (txtCmd.Text.Equals("PWD"))
    {
        logger.Logs.Clear();
        await ftpClient.GetPresentWorkingDirectoryAsync();
        return;
    }
    ...
}
```

Internally, `GetPresentWorkingDirectoryAsync` sends PWD to the FTP server over the socket:

```
public async Task GetPresentWorkingDirectoryAsync()
{
    if (!IsBusy)
    {
        ftpCommand = FtpCommand.PresentWorkingDirectory;
        logger.AddLog("FTPClient -> PWD\r\n");
        await FtpCommandSocket.SendData("PWD\r\n");
    }
}
```

When this procedure is successful, the FTP server sends the response with the path of the present working directory, the FTP client is notified and the `FtpPresentWorkingDirectoryReceived` event is raised. Using the event arguments, the client can get the information about the path of the present working directory (as shown in **Figure 5**):

```
void ftpClient_FtpPresentWorkingDirectoryReceived(object sender,
    FtpPresentWorkingDirectoryEventArgs e)
{
    // Handle PresentWorkingDirectoryReceived event to show some
    // prompt or message to user
}
```

To change to some other directory, the FTP client can use the Change Working Directory (CWD) command. Note that CWD only works to change to a subdirectory in the current working directory or to its parent directory. When the user is in the root directory, he can't navigate backward and CWD will throw an error in response.

To change the working directory, I issue the CWD command in the command textbox, and in the Send Command button's tap event I call the `ChangeWorkingDirectoryAsync` method:

```
async private void btnFtp_Tap(object sender,
    System.Windows.Input.GestureEventArgs e)
{
    ...
    if (txtCmd.Text.StartsWith("CWD"))
    {
        logger.Logs.Clear();
        await ftpClient.ChangeWorkingDirectoryAsync(
            txtCmd.Text.Split(new char[] { ' ' },
                StringSplitOptions.RemoveEmptyEntries)[1]);
        return;
    }
    ...
}
```

Internally, `ChangeWorkingDirectoryAsync` issues the CWD command to the FTP server:

```
public async Task ChangeWorkingDirectoryAsync(String RemoteDirectory)
{
    if (!IsBusy)
    {
        this.RemoteDirectory = RemoteDirectory;
        ftpCommand = FtpCommand.ChangeWorkingDirectory;
        logger.AddLog(String.Format("FTPClient -> CWD {0}\r\n", RemoteDirectory));
        await FtpCommandSocket.SendData(String.Format("CWD {0}\r\n", RemoteDirectory));
    }
}
```

If the user wants to navigate backward, he can send double dots ".." as a parameter to this procedure. When the change is successful, the client is notified and the `FtpDirectoryChangedSucceeded` event is raised, which can be seen in **Figure 6**. If CWD fails and sends an error in response, the client is notified of that failure and the `FtpDirectoryChangedFailed` event is raised:

```
void ftpClient_FtpDirectoryChangedSucceeded(object sender,
    FtpDirectoryChangedEventArgs e)
{
    // Handle DirectoryChangedSucceeded event to show
    // some prompt or message to user
}

void ftpClient_FtpDirectoryChangedFailed(object sender,
    FtpDirectoryChangedEventArgs e)
{
    // Handle DirectoryChangedFailed event to show
    // some prompt or message to user
}
```

Adding Support for Passive Commands

Now it's time to provide support for passive commands, because I want the FTP server to manage all the data connections for transferring data to and from the server. In passive mode, any command that needs to transfer data must use a data connection initiated by the FTP server. The FTP server will create a data connection and send the socket information for that data connection so the client can connect to it and perform some data-transfer operations.

Passive mode is temporary in nature. Before sending a command to the FTP server that will send or receive data in response, the



Figure 10 Sending LIST Command to the FTP Server



Figure 11 Displaying File System Objects in Response to LIST Command



AMPLIFY YOUR KNOWLEDGE

5 Days. 4 Events.
20 Tracks.
175+ Sessions.
The Ultimate IT and
Developer Line-up.

live360events.com

Live! 360 is back to turn your conference experience up to 11. Spanning 5 days at the Royal Pacific Resort in sunny Orlando, Live! 360 gives you the ultimate IT and Developer line-up, offering hundreds of sessions on the most relevant topics affecting your business today.

**REGISTER TODAY
SAVE \$400!**

**SUPER EARLY BIRD SAVINGS
END SEPTEMBER 12**

Use Promo Code LIVESEP1



Scan the QR code to register or for more event details.



Figure 12 The UploadFileAsync Method

```
async private void btnFtp_Tap(object sender, System.Windows.Input.GestureEventArgs e)
{
    ...
    if (txtCmd.Text.StartsWith("STOR"))
    {
        logger.Logs.Clear();
        String Filename = txtCmd.Text.Split(new char[] { ' ', '/' },
            StringSplitOptions.RemoveEmptyEntries).Last();
        StorageFile file =
            await Windows.Storage.ApplicationData.Current.LocalFolder.GetFileAsync(
                txtCmd.Text.Split(new char[] { ' ' },
                    StringSplitOptions.RemoveEmptyEntries)[1]);
        await ftpClient.UploadFileAsync(await file.OpenStreamForReadAsync(),
            "video2.mp4");
        return;
    }
    ...
}
```

client has to tell the server to go into passive mode—and it must do so for every command sent. In a nutshell, for every data transfer, two commands are always fired—passive and then the command itself.

The FTP server is told to prepare for passive mode via the PASV command. In response, it sends the IP address and port number of the data connection in an encoded format. The response is decoded and the client then prepares the data connection with the FTP server using the decoded IP address and port number. Once the data operation is complete, this data connection is closed and dissolved, so that it can't be used again. This happens every time the PASV command is sent to the FTP server.

Decoding the PASV Command Response The response to the PASV command looks like this:

```
227 Entering Passive Mode (192,168,33,238,255,167)
```

The response has data channel information contained within it. Based on this response, I have to formulate the socket address—the IP address and data port the FTP server is using for the data-transfer operation. Here are the steps to calculate the IP address and port, as shown in **Figure 7**:

- The first four integer groups form the IPV4 address of the FTP server; that is, 192.168.33.238.
- The remaining integers comprise the data port. You left-shift the fifth integer group with 8 and then perform Bitwise OR operation with the sixth integer group. The final value gives you the port number where the data channel will be available.

Figure 8 shows the code that parses the response and extracts the IP address and port number of the data channel's endpoint. The method PrepareDataChannelAsync receives the raw FTP response for the PASV command. Sometimes the response string can be changed on the FTP server to include some other parameters, but essentially the response sends just the IP address and port number.

More Commands: List, Type, STOR and RETR

Listing the Directory Contents To list the contents of the current working directory, I send the LIST command over the command channel to the FTP server. However, the FTP server will send its response to this command over the data channel, so I must first send the PASV command in order to create the data connection over which the LIST command's response will be sent. The directory listing format will be based on the OS on which the FTP server is installed. That is, if the

FTP server OS is Windows, the directory listing will be received in the Windows directory listing format, and if the OS is Unix-based, the directory listing will be received in Unix format.

To list the directory contents of the present working directory, I issue the LIST command in the command textbox and in the Send Command button's tap event, I call the GetDirectoryListingAsync method:

```
async private void btnFtp_Tap(object sender,
    System.Windows.Input.GestureEventArgs e)
{
    ...
    if (txtCmd.Text.Equals("LIST"))
    {
        logger.Logs.Clear();
        await ftpClient.GetDirectoryListingAsync();
        return;
    }
    ...
}
```

Internally, GetDirectoryListingAsync sends the PASV command to the FTP server over the socket:

```
public async Task GetDirectoryListingAsync()
{
    if (!IsBusy)
    {
        fileListingData = null;
        IsBusy = true;
        ftpCommand = FtpCommand.Passive;
        logger.AddLog("FTPClient -> PASV\r\n");
        await FtpCommandSocket.SendData("PASV\r\n");
    }
}
```

Once the endpoint information is received for the PASV command, the data connection is created and the LIST command is then issued to the FTP server, as shown in **Figure 9**.

Figure 13 Processing the STOR Command in the Socket's DataReceived Event

```
async void FtpClientSocket_DataReceived(
    object sender, DataReceivedEventArgs e)
{
    String Response = System.Text.Encoding.UTF8.GetString(
        e.GetData(), 0, e.GetData().Length);
    ...
    IsBusy = true;
    DataWriter dataWriter = new DataWriter(
        FtpDataChannel.OutputStream);
    byte[] data = new byte[32768];
    while (!(await ftpFileInfo.LocalFileStream.ReadAsync(
        data, 0, data.Length)).Equals(0))
    {
        dataWriter.WriteBytes(data);
        await dataWriter.StoreAsync();
        RaiseFtpFileTransferProgressedEvent(
            Convert.ToInt32(data.Length), true);
    }
    await dataWriter.FlushAsync();
    dataWriter.Dispose();
    dataWriter = null;
    FtpDataChannel.Dispose();
    FtpDataChannel = null;
    ...
    await PrepareDataChannelAsync(Response);
    ftpPassiveOperation = FtpPassiveOperation.FileUpload;
    logger.AddLog(String.Format("FTPClient -> STOR {0}\r\n",
        ftpFileInfo.RemoteFile));
    await FtpCommandSocket.SendData(String.Format("STOR {0}\r\n",
        ftpFileInfo.RemoteFile));
    ...
}
```

Modern Apps LIVE!

MODERN APPS FROM START TO FINISH

ORLANDO

NOVEMBER 18-22, 2013

LOEWS ROYAL PACIFIC RESORT
AT UNIVERSAL

THE FUTURE OF SOFTWARE DEVELOPMENT IS BACK!

Critics raved about the Modern Apps Live! performance in March, so it's back for an encore performance! Presented in partnership with Magenic, Modern Apps Live! brings Development Managers, Software Architects and Development Leads together to break down the latest and greatest techniques in low-cost, high-value application development.

REGISTER TODAY TO SAVE \$4.00!

Super Early Bird Savings End September 12. | Use Promo Code MALSEP1



Presented in partnership with

Magenic



IT EVENTS WITH PERSPECTIVE

Four co-located events;
one low price!

- Modern Apps Live!
- SharePoint Live!
- Visual Studio Live! Orlando
- SQL Server Live!

live360events.com



Scan the QR code for more information on Modern Apps Live!

SUPPORTED BY
Microsoft

 Visual Studio

msdn
magazine

Visual Studio
MAGAZINE

PRODUCED BY

 1105 MEDIA



Figure 14 Uploading a File to the FTP Server



Figure 15 Successful File Upload

When the directory listing is received over the data channel, the `FtpDirectoryListed` event is raised with the list of files and directories in its event arguments (Figure 10 shows sending the PASV and LIST commands, and Figure 11 displays the output of directory listing):

```
void ftpClient_FtpDirectoryListed(object sender, FtpDirectoryListedEventArgs e)
{
    foreach (String filename in e.GetFileNames())
    {
        // Handle the name of filenames in current working directory
    }

    foreach (String directory in e.GetDirectories())
    {
        // Handle the name of directories in current working directory
    }
}
```

Describing the Data Format The TYPE command is used to describe the data format in which the file's data will be received or sent. It doesn't require PASV mode. I use the "I" format with the TYPE command to signify an Image-type data transmission. TYPE is generally used when storing or retrieving files over a data connection. This tells the FTP server that the transmission will happen in binary mode rather than in textual or some data-structural mode.

Figure 16 The DownloadFileAsync Method

```
async private void btnFtp_Tap(
    object sender, System.Windows.Input.GestureEventArgs e)
{
    ...
    if (txtCmd.Text.StartsWith("RETR"))
    {
        logger.Logs.Clear();
        String filename = txtCmd.Text.Split(new char[] { ' ', '/' },
            StringSplitOptions.RemoveEmptyEntries).Last();
        StorageFile file =
            await Windows.Storage.ApplicationData.Current.LocalFolder.
                CreateFileAsync(
                    txtCmd.Text.Split(new char[] { ' ' }, StringSplitOptions.
                        RemoveEmptyEntries)[1],
                    CreationCollisionOption.ReplaceExisting);
        await ftpClient.DownloadFileAsync(
            await file.OpenStreamForWriteAsync(), filename);
        return;
    }
    ...
}
```

For other modes that can be used with the TYPE command, refer to RC 959 for FTP.

Storing a File on the FTP Server To store the contents of a file (located on a device) to an FTP server, I send the STOR command along with the name of the file (and extension) the FTP server will use to create and save the file. The transmission of the file contents will be performed on a data connection so, before sending STOR, I'll query the endpoint details from the FTP server using the PASV command. Once the endpoint is received, I'll send STOR, and when I receive the response to it, I'll send the file contents in binary form to the FTP server over the data connection.

The STOR command can be sent by calling the `UploadFileAsync` method, as shown in Figure 12. This method takes two parameters—the Stream object of the local file and the name of the file on the FTP server as a String.

Internally, the `UploadFileAsync` method issues a PASV command to the FTP server to fetch the data channel endpoint information:

```
public async Task UploadFileAsync(
    System.IO.Stream LocalFileStream, String RemoteFilename)
{
    if (!IsBusy)
    {
        ftpFileInfo = null;
        IsBusy = true;
        ftpFileInfo = new FtpFileOperationInfo(LocalFileStream, RemoteFilename, true);
        ftpCommand = FtpCommand.Type;
        logger.AddLog("FTPClient -> TYPE I\r\n");
        await FtpCommandSocket.SendData("TYPE I\r\n");
    }
}
```

As shown in Figure 13, in the PASV command's response within the `DataReceived` event, the STOR command is issued after the data channel is created and opened; the data transmission then starts from client to server.

While the file is uploading, the client is notified of the upload progress via `FtpFileTransferProgressed`:

```
void ftpClient_FtpFileTransferProgressed(
    object sender, FtpFileTransferProgressedEventArgs e)
{
    // Update the UI with some progressive information
    // or use this to update progress bar
}
```

If the file-uploading operation completes successfully, the `FtpFileUploadSucceeded` event is raised. If not, the `FtpFileUploadFailed` event is raised with the failure reason in its argument:

```
void ftpClient_FtpFileUploadSucceeded(
    object sender, FtpFileTransferEventArgs e)
{
    // Handle UploadSucceeded Event to show some
    // prompt or message to user
}

void ftpClient_FtpFileUploadFailed (
    object sender, FtpFileTransferEventArgs e)
{
    // Handle UploadFailed Event to show some
    // prompt or message to user
}
```

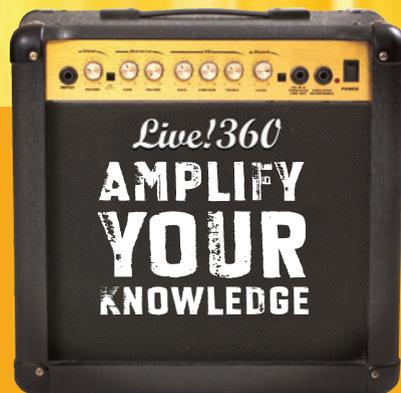
Figure 14 and Figure 15 display the process of uploading a file to the FTP server.

Retrieving a File from an FTP Server To retrieve the contents of a file from an FTP server, I send the RETR command along with the name of the file and its extension to the FTP server (assuming the file is on the server in the current directory). The transmission of the file contents will use a data connection, so, before sending RETR, I'll

SharePoint LIVE!

TRAINING FOR COLLABORATION

ORLANDO
NOVEMBER 18-22, 2013
LOEWS ROYAL PACIFIC RESORT
AT UNIVERSAL



IT EVENTS WITH PERSPECTIVE

Four co-located events;
one low price!

- SharePoint Live!
- Visual Studio Live! Orlando
- SQL Server Live!
- Modern Apps Live!

live360events.com

COLLABORATE AND LISTEN

Whether you've implemented SharePoint 2013, or you're still using and supporting older versions, SharePoint Live! is THE place to gather and learn how to customize, deploy and maintain SharePoint Server and SharePoint foundation to maximize business value.

REGISTER TODAY TO SAVE \$4.00!

Super Early Bird Savings End September 12.

Use Promo Code SPSEP1



Scan the QR code for more information on SharePoint Live!

SUPPORTED BY



PRODUCED BY



Figure 17 Processing the RETR Command in the Socket's DataReceived Event

```

async void FtpClientSocket_DataReceived(
    object sender, DataReceivedEventArgs e)
{
    String Response = System.Text.Encoding.UTF8.GetString(
        e.GetData(), 0, e.GetData().Length);

    ...

    IsBusy = true;
    DataReader dataReader = new DataReader(FtpDataChannel.InputStream);
    dataReader.InputStreamOptions = InputStreamOptions.Partial;
    while (!(await dataReader.LoadAsync(32768)).Equals(0))
    {
        IBuffer databuffer = dataReader.DetachBuffer();
        RaiseFtpFileTransferProgressedEvent(databuffer.Length, false);
        await ftpFileInfo.LocalFileStream.WriteAsync(
            databuffer.ToArray(), 0, Convert.ToInt32(databuffer.Length));
    }
    await ftpFileInfo.LocalFileStream.FlushAsync();
    dataReader.Dispose();
    dataReader = null;
    FtpDataChannel.Dispose();
    FtpDataChannel = null;

    ...

    await PrepareDataChannelAsync(Response);
    ftpPassiveOperation = FtpPassiveOperation.FileDownload;
    logger.AddLog(String.Format("FTPClient -> RETR {0}\r\n",
        ftpFileInfo.RemoteFile));
    await FtpCommandSocket.SendData(String.Format("RETR {0}\r\n",
        ftpFileInfo.RemoteFile));

    ...
}
}

```

query the endpoint details from the FTP server using the PASV command. Once the endpoint is received, I'll send the RETR command, and after receiving the response to it, I'll retrieve the file contents in binary form from the FTP server over the data connection.

As shown in **Figure 16**, the RETR command can be sent by calling the DownloadFileAsync method. This method takes two parameters—the Stream object of the local file for saving the content and the name of the file on the FTP server as a String.



Figure 18 Downloading a File from the FTP Server



Figure 19 Successful File Download

Internally, the DownloadFileAsync method issues a PASV command to the FTP server to fetch the data channels endpoint information:

```

public async Task DownloadFileAsync(
    System.IO.Stream LocalFileStream, String RemoteFilename)
{
    if (!IsBusy)
    {
        ftpFileInfo = null;
        IsBusy = true;
        ftpFileInfo = new FtpFileOperationInfo(
            LocalFileStream, RemoteFilename, false);
        ftpCommand = FtpCommand.Type;
        logger.AddLog("FTPClient -> TYPE I\r\n");
        await FtpCommandSocket.SendData("TYPE I\r\n");
    }
}

```

As shown in **Figure 17**, in the PASV command's response inside the DataReceived event, the RETR command is issued after the data channel is created and opened; then the data transmission starts, running from server to client.

While the file is downloading, the client is notified of the download progress via FtpFileTransferProgressed:

```

void ftpClient_FtpFileTransferProgressed(object sender,
    FtpFileTransferProgressedEventArgs e)
{
    // Update the UI with some progressive information or use
    // this to update progress bar
}

```

If the file downloading operation completes successfully, the FtpFileDownloadSucceeded event is raised. If not, the FtpFileDownloadFailed event is raised with the failure reason in its argument:

```

void ftpClient_FtpFileDownloadSucceeded(object sender,
    FtpFileTransferFailedEventArgs e)
{
    // Handle DownloadSucceeded event to show some prompt
    // or message to user
}

void ftpClient_FtpFileDownloadFailed(object sender,
    FtpFileTransferFailedEventArgs e)
{
    // Handle UploadFailed Event to show some prompt
    // or message to user
}

```

Figure 18 and **Figure 19** display the process of downloading a file from the FTP server.

Wrapping Up

Note that you can provide a URI association to an app implementing FTP, so that other apps can also access the FTP service and request data from the FTP server. You'll find more information about this on the Windows Phone Dev Center page, "Auto-launching apps using file and URI associations for Windows Phone 8," at bit.ly/XeAaZ8, and sample code at bit.ly/15x400y.

The code I've written for my FTP library and client app for Windows Phone is fully supported on Windows 8.x, as I haven't used any API that isn't compatible with Windows 8.x. You can either recompile the code for Windows 8.x, or put it in a Portable Class Library (PCL) that can target both platforms. ■

UDAY GUPTA is senior engineer - product development for Symphony Teleca Corp. (India) Pvt Ltd. He has experience in many .NET technologies, especially in Windows Presentation Foundation (WPF), Silverlight, Windows Phone and Windows 8. Most of his time is spent in coding, gaming, learning new things and helping others.

THANKS to the following technical experts for reviewing this article: Tony Champion (Champion DS) and Andy Wigley (Microsoft)

SQL Server® LIVE!

TRAINING FOR DBAs AND IT PROS

ORLANDO
NOVEMBER 18-22, 2013
LOEWS ROYAL PACIFIC RESORT
AT UNIVERSAL



FINE TUNE YOUR DATA

After 5 days of workshops, deep dives and breakout sessions, SQL Server Live! will leave you with finely tuned skills to solve your biggest data challenges. When we say SQL Server Live! provides comprehensive education and knowledge share on SQL Server database management, data warehouse/BI model design, Big Data analytics, performance tuning, troubleshooting and coding against SQL Server – we mean it.

REGISTER TODAY TO SAVE \$4.00!

Super Early Bird Savings End September 12.

Use Promo Code SQLSEP1



IT EVENTS WITH PERSPECTIVE

Four co-located events;
one low price!

- SQL Server Live!
- Visual Studio Live! Orlando
- SharePoint Live!
- Modern Apps Live!

live360events.com



Scan the QR code for more information on SQL Server Live!



Multi-Swarm Optimization

Multi-swarm optimization (MSO) is a technique for estimating the solution to difficult or impossible numerical problems. It's a variation of particle swarm optimization (see my article on the subject at msdn.microsoft.com/magazine/hh335067). Regular particle swarm optimization models flocking behavior, such as that seen in groups of birds and schools of fish. MSO extends particle swarm optimization by using several swarms of simulated particles rather than a single swarm.

MSO can be applied to several machine-learning scenarios, such as estimating the weights and bias values for an artificial neural network or estimating the weights of weak learners in ensemble classification and prediction. MSO is a meta-heuristic, meaning that the technique is really a set of design principles and guidelines that can be used to construct a specific algorithm to solve a specific optimization problem.

A good way to understand multi-swarm optimization is to examine the demo program in **Figure 1** and the graph in **Figure 2**. The demo program is estimating the solution to a standard benchmark numerical optimization problem called Rastrigin's function. The goal is to find the values of x_0 and x_1 that minimize the function:

$$f(x_0, x_1) = 20 + \sum_{i=0}^1 (x_i^2 - 10 \cos 2\pi x_i)$$

The function has a known solution of $f = 0.0$ when $x_0 = 0.0$ and $x_1 = 0.0$, so the use of MSO isn't really necessary in this situation. The graph in **Figure 2** shows Rastrigin's function. Though the function has many peaks and valleys representing false solutions, there's only one global minimum at the center of the image. The multiple close-but-not-quite solutions of Rastrigin's function are deliberate and are designed to cause trouble for optimization algorithms.

The demo program in **Figure 1** begins by setting up some input parameters for the MSO algorithm. There are three swarms, and each swarm has four particles. In most numerical optimization problems, the range of possible values is limited in some way. Here, the search space is initially limited to x values between -100.0 and $+100.0$.

The demo program initializes each of the 12 particles to random (x_0, x_1) values. Each pair of values represents a particle position that can also be interpreted as a possible solution. The value of Rastrigin's function at each position is called the cost of the position, to suggest that the goal is to minimize the function. After

initialization, the best particle (the one with the smallest cost) is zero-based index particle 0 in swarm 2. That particle has position $[-40.57, 28.54]$ and associated cost 2498.93.

Multi-swarm optimization is an iterative process. The demo program sets an arbitrary maximum loop value of 150. The MSO algorithm then searches for better solutions, keeping track of the best solution found by any of the 12 particles. At the end of 150 iterations, the best solution found was $f = 0.000043$ at $x_0 = -0.0003$, $x_1 = 0.0004$, which is very close to but not quite the actual solution. The demo program can in fact find the actual solution by setting the `maxLoop` variable to 500, but it's important to remember that in most realistic problem scenarios you won't know if MSO has found the optimal solution.

This article assumes you have at least intermediate-level programming skills but doesn't assume you know anything about multi-swarm optimization. The demo program is coded in C# but you shouldn't have too much trouble refactoring to another language. In order to keep the size of the code small and the main ideas clear, I've removed most of the normal error checking from the demo code. The demo is too long to present in its entirety in this article, but the entire source code is available at archive.msdn.microsoft.com/mag201309TestRun.

Overall Program Structure

The structure of the demo program, with some minor edits and most of the `WriteLine` statements removed, is presented in **Figure 3**.

To create the demo program I used Visual Studio 2012. The demo has no significant dependencies, and any version of Visual Studio should work. I selected the C# console application template and named the project `MultiSwarm`. After Visual Studio loaded the code template, in the Solution Explorer window I renamed the `Program.cs` file to `MultiSwarmProgram.cs` and Visual Studio automatically renamed the program class for me. At the top of the source code I deleted all unneeded namespace references, leaving just the reference to the `System` namespace.

The demo program defines a public-scope `Cost` method, which is Rastrigin's function:

```
public static double Cost(double[] position)
{
    double result = 0.0;
    for (int i = 0; i < position.Length; ++i) {
        double xi = position[i];
        result += (xi * xi) - (10 * Math.Cos(2 * Math.PI * xi)) + 10;
    }
    return result;
}
```

Code download available at archive.msdn.microsoft.com/mag201309TestRun.

The Cost method accepts a single array parameter that represents a particle's position, which is a possible solution. In most machine-learning scenarios, the Cost function you're trying to minimize represents some form of error and will require additional parameters, such as a training data source. The Cost function is usually the performance bottleneck for MSO, so you should code it to be as efficient as possible.

The Multiswarm class encapsulates the MSO algorithm. Classes Particle and Swarm are helper classes for class Multiswarm. An alternative design in languages that support nested class definitions is to define classes Particle and Swarm inside class Multiswarm.

The demo program begins by setting up five input parameters:

```
int dim = 2;
double minX = -100.0;
double maxX = 100.0;
int numParticles = 4;
int numSwarms = 3;
```

Variable dim represents the number of dimensions in the problem to be solved. Because Rastrigin's function accepts x_0 and x_1 , the value of dim is set to 2. Multi-swarm optimization is well-suited for problems with any number of dimensions. Variables minX and maxX constrain the search to a limited range of values. The values for minX and maxX will vary from problem to problem. Variable numParticles defines the number of particles that are in each swarm. Variable numSwarms defines the total number of swarms. In general, larger values of numParticles and numSwarms produce more accurate solutions at the expense of performance.

Next, the primary multi-swarm object is instantiated and the MSO solving algorithm is called:

```
Multiswarm ms = new Multiswarm(numSwarms, numParticles, dim, minX, maxX);
Console.WriteLine("\nInitial multiswarm:");
Console.WriteLine(ms.ToString());
int maxLoop = 150;
ms.Solve(maxLoop);
```

The demo calls a collection of particles a "swarm," and a collection of swarms a "multi-swarm." Instead of this naming scheme, some research literature calls a collection of particles a "sub-swarm" and a collection of sub-swarms a "swarm." The Multiswarm object is instantiated using the previously defined input parameters. Variable maxLoop holds the maximum number of times the main solving algorithm loop will iterate. In general, larger values of maxLoop will produce more accurate solutions at the expense of performance.

The Solve method iteratively searches for the best solution to Rastrigin's function. Notice that the Cost function is defined externally to the Multiswarm object. In most situations, MSO code

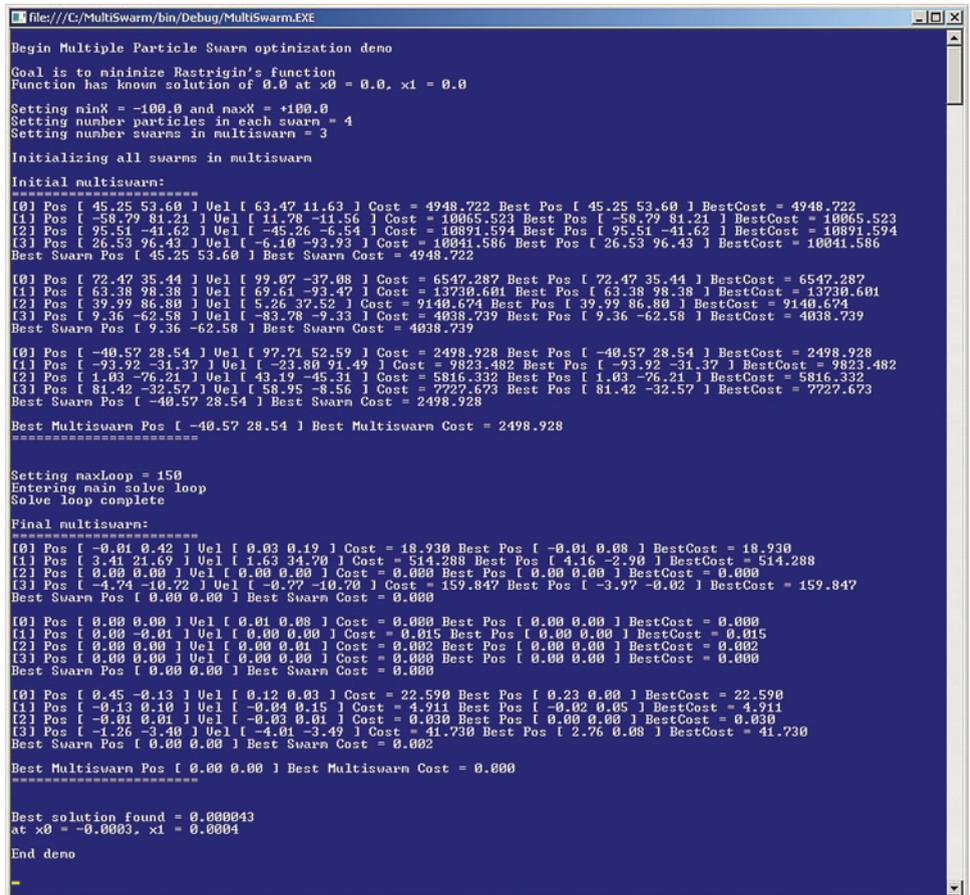


Figure 1 Multi-Swarm Optimization Demo

is integrated into a software system rather than implemented as a class library DLL, which requires you to pass the Cost function into the Multiswarm object (via a delegate, for example) or use an interface-design approach to define a programming contract.

After the Solve method finishes executing, the final state of the Multiswarm object is displayed, and the best solution found by any particle in any swarm in the multi-swarm is explicitly displayed:

```
Console.WriteLine("\nFinal multiswarm:");
Console.WriteLine(ms.ToString());

Console.WriteLine("\nBest solution found = " +
ms.bestMultiCost.ToString("F6"));
Console.WriteLine("at x0 = " + ms.bestMultiPos[0].ToString("F4"));
Console.WriteLine(" , x1 = " + ms.bestMultiPos[1].ToString("F4"));
```

Particles

The Particle class has six members:

```
static Random ran = new Random();
public double[] position;
public double[] velocity;
public double cost;
public double[] bestPartPos;
public double bestPartCost;
```

I prefer to use public scope for simplicity but you may want to use private scope along with get and set property methods. The Random object is used by the constructor to initialize a Particle object to a random position. The array named position represents the position of a particle. Array velocity represents the speed and direction for

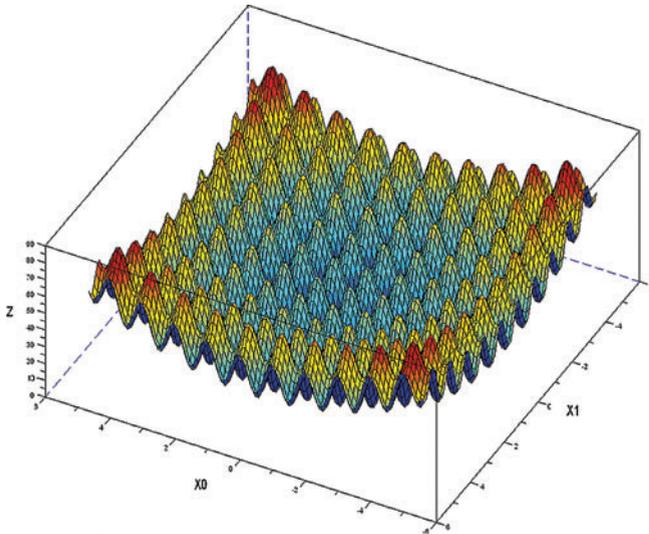


Figure 2 Rastrigin's Function

a particle. For example, suppose that a particle is at position [12.0, 24.0] and the velocity is [5.0, 0.0]. This can be interpreted to mean that during the next time increment the particle will move 5.0 units along the x0 dimension and 0.0 units along the x1 dimension. After the particle moves, its new position will be [17.0, 24.0].

Variable cost holds the value of the Cost function at the current position. Variable bestPartCost holds the best (smallest) cost value that a particle ever encountered, and variable bestPartPos is the position where the best-known cost was found.

The Particle constructor is defined in **Figure 4**.

The constructor allocates space for the position, velocity and bestPartPos arrays based on the number of problem dimensions. Each position and velocity cell is assigned a random value between minX and maxX. The cost of the initial position is computed. The particle's best-known position and cost are set to the initial position and cost.

A significant alternative approach is to assign non-random initial positions to each particle. Some MSO research literature suggests that assigning particles in different swarms to different regions of the search space is superior to a random-assignment approach. For example, if you had two swarms with 10 particles each, the 10 particles in the first swarm could be assigned to positions with x-values between -100.0 and 0.0, and the 10 particles in the second swarm to positions with x-values between 0.0 and +100.0. I'm not entirely convinced by these research results, however, and simple random particle position assignment has worked well for me in practice.

Swarms

A Swarm is a collection of particles. The Swarm class has three members:

```
public Particle[] particles;
public double[] bestSwarmPos;
public double bestSwarmCost;
```

Naming can be a bit tricky with MSO. I name the array of Particle objects in the Swarm class as "particles," but you might want to use the name "swarm" instead. Member variable bestSwarmCost holds the best (smallest) cost found by any of the particles in the swarm during algorithm execution. Array bestSwarmPos holds the position where this best swarm-member cost was found.

The Swarm constructor is shown in **Figure 5**.

The Swarm constructor allocates space, then calls the Particle constructor numParticles times to generate random-position particles. As each particle is created, it's checked to see if it has the best cost of any of the particles in the swarm.

The Multiswarm Class

A multi-swarm is a collection of swarms. The top-level Multiswarm class has seven members:

```
public Swarm[] swarms;
public double[] bestMultiPos;
public double bestMultiCost;
public int dim;
public double minX;
public double maxX;
static Random ran = new Random(0);
```

Array swarms holds each Swarm object, each of which is a collection of Particle objects. So swarms[2].particles[3].position[0] represents the x0-value for particle 3 in swarm 2.

Member variable bestMultiCost holds the best cost found by any particle in any swarm during algorithm execution. Array

Figure 3 Multi-Swarm Demo Program Structure

```
using System;
namespace MultiSwarm
{
    class MultiSwarmProgram
    {
        static void Main(string[] args)
        {
            try
            {
                Console.WriteLine(
                    "\nBegin Multiple Particle Swarm optimization demo\n");

                int dim = 2;
                double minX = -100.0;
                double maxX = 100.0;

                int numParticles = 4; // Particles in each swarm
                int numSwarms = 3; // Swarms in multi-swarm

                Multiswarm ms = new Multiswarm(numSwarms, numParticles, dim, minX, maxX);
                Console.WriteLine("\nInitial multiswarm:");
                Console.WriteLine(ms.ToString());

                int maxLoop = 150;
                ms.Solve(maxLoop);

                Console.WriteLine("\nFinal multiswarm:");
                Console.WriteLine(ms.ToString());

                Console.WriteLine("\nBest solution found = " +
                    ms.bestMultiCost.ToString("F6"));
                Console.WriteLine("at x0 = " + ms.bestMultiPos[0].ToString("F4"));
                Console.WriteLine("x1 = " + ms.bestMultiPos[1].ToString("F4"));

                Console.WriteLine("\nEnd demo\n");
                Console.ReadLine();
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
                Console.ReadLine();
            }
        }

        public static double Cost(double[] position) { . . . }
    } // Program

    public class Particle { . . . }
    public class Swarm { . . . }
    public class Multiswarm { . . . }
} // ns
```

bestMultiPos holds the associated position where the best global cost was found. Member variables dim, minX and maxX are stored for convenience so their values can be used by class methods without being passed as parameters.

Recall that class Particle has a Random object named ran that's used to generate initial random positions. Class Multiswarm has a different Random object that's used by the MSO algorithm to insert pseudo-random behavior during the solve process.

The Multiswarm constructor is listed in **Figure 6**.

The key operation in MSO is computing a new velocity for a particle.

After allocating arrays and saving input parameter values, the Multiswarm constructor calls the Swarm constructor numSwarms times. As each swarm is created, the best cost of any particle within that swarm is checked to see if it's a global best cost. If so, that cost and its associated position are stored into bestMultiCost and bestMultiPos, respectively.

The MSO Algorithm

In very high-level pseudo-code, the basic MSO algorithm is:

```

loop maxLoop times
  for each swarm
    for each particle
      compute new velocity
      use velocity to update position
      check if a new best cost has been found
    end for
  end for
end loop

```

The key operation in MSO is computing a new velocity for a particle. A new velocity for a given particle is influenced by the current velocity, the current position, the best-known position of the particle, the best-known position of any particle in the same swarm as the particle, and the best-known position of any particle in any swarm. In math terms, the new velocity is:

$$\mathbf{v}(t+1) = w * \mathbf{v}(t) + (c1 * r1) * (\mathbf{p}(t) - \mathbf{x}(t)) + (c2 * r2) * (\mathbf{s}(t) - \mathbf{x}(t)) + (c3 * r3) * (\mathbf{m}(t) - \mathbf{x}(t))$$

After a new velocity has been computed, a particle's new position is:

$$\mathbf{x}(t+1) = \mathbf{x}(t) + \mathbf{v}(t+1)$$

Bear with me for a moment. The computation is much simpler than it first appears. The term $\mathbf{v}(t+1)$ means the velocity at time $t+1$, in other words, the new velocity. Term $\mathbf{v}(t)$ is the current velocity. Term $\mathbf{x}(t)$ is the current position. Notice that \mathbf{x} and \mathbf{v} are in bold, which indicates they're vectors such as [12.0, 25.0] rather than single values.

Term $\mathbf{p}(t)$ is a particle's best-known position. Term $\mathbf{s}(t)$ is the best position of any particle in the particle's swarm. Term $\mathbf{m}(t)$ is the best position of any particle in any swarm.

Term w is a constant called the inertia factor. Terms $c1$, $c2$ and $c3$ are constants that establish a maximum change for each component of the new velocity. Terms $r1$, $r2$, and $r3$ are random values between 0 and 1 that provide a randomization effect to each velocity update.

The new velocity computation is probably best explained by example. Suppose a particle is currently at [12.0, 24.0] and its current velocity is [-1.0, -3.0]. Also, the best-known position of the particle is [8.0, 10.0], the best-known position of any particle in the swarm is [7.0, 9.0], and the best-known position of any particle in any swarm is [5.0, 6.0]. And suppose that constant w has value 0.7, constants $c1$ and $c2$ are both 1.4, and constant $c3$ is 0.4. Finally, suppose random values $r1$, $r2$ and $r3$ are all 0.2.

The new velocity of the particle is shown in **Figure 7**.

And so, according to **Figure 7**, the particle's new position is:

$$\mathbf{x}(t+1) = [12.0, 24.0] + [-4.1, -12.6] = [7.9, 11.4]$$

Assuming the optimal solution is [0.0, 0.0], as is the case with Rastrigin's function, notice that the particle has moved from its original position to a new position that's closer to the optimal solution.

The inertia term in $\mathbf{v}(t+1)$ encourages a particle to continue moving in its current direction. The $\mathbf{p}(t)$ term encourages a particle to move toward its historical best-known position. The $\mathbf{s}(t)$ term encourages a particle to move toward the best-known position found by any of the particle's swarm-mates. The $\mathbf{m}(t)$ term encourages a particle to move toward the best-known position found by any particle in any swarm.

Constants $c1$, $c2$, and $c3$ are sometimes called the cognitive, social, and global weights. Those constants, along with random values $r1$, $r2$, and $r3$, and the inertia weight w , determine how much each term influences the motion of a particle. Some research in regular particle swarm optimization suggests reasonable values for w , $c1$, and $c2$ are 0.729, 1.49445, and 1.49445, respectively. There's little research on the $c3$ constant in MSO, but I typically use 0.3645 (half of the inertia weight), and this has worked well for me in practice.

Death and Immigration

There are several fascinating ways to modify the basic MSO algorithm. One possibility is to essentially kill a randomly selected

Figure 4 The Particle Constructor

```

public Particle(int dim, double minX, double maxX)
{
    position = new double[dim];
    velocity = new double[dim];
    bestPartPos = new double[dim];
    for (int i = 0; i < dim; ++i) {
        position[i] = (maxX - minX) * ran.NextDouble() + minX;
        velocity[i] = (maxX - minX) * ran.NextDouble() + minX;
    }
    cost = MultiSwarmProgram.Cost(position);
    bestPartCost = cost;
    Array.Copy(position, bestPartPos, dim);
}

```

Figure 5 The Swarm Constructor

```

public Swarm(int numParticles, int dim, double minX, double maxX)
{
    bestSwarmCost = double.MaxValue;
    bestSwarmPos = new double[dim];
    particles = new Particle[numParticles];
    for (int i = 0; i < particles.Length; ++i) {
        particles[i] = new Particle(dim, minX, maxX);
        if (particles[i].cost < bestSwarmCost) {
            bestSwarmCost = particles[i].cost;
            Array.Copy(particles[i].position, bestSwarmPos, dim);
        }
    }
}

```

Figure 6 The Multiswarm Constructor

```
public Multiswarm(int numSwarms, int numParticles, int dim,
    double minX, double maxX)
{
    swarms = new Swarm[numSwarms];
    bestMultiPos = new double[dim];
    bestMultiCost = double.MaxValue;
    this.dim = dim;
    this.minX = minX;
    this.maxX = maxX;

    for (int i = 0; i < numSwarms; ++i)
    {
        swarms[i] = new Swarm(numParticles, dim, minX, maxX);
        if (swarms[i].bestSwarmCost < bestMultiCost)
        {
            bestMultiCost = swarms[i].bestSwarmCost;
            Array.Copy(swarms[i].bestSwarmPos, bestMultiPos, dim);
        }
    }
}
```

particle every now and then, and then give birth to a new particle. The demo program uses this death-birth modification. The first few lines of the Solve method are shown in Figure 8.

The method generates a random value between 0 and 1 and stores it into *p*. If the random value is less than 0.005, the current particle is re-instantiated by calling the Particle constructor, effectively killing the current particle and giving birth to a new particle at a random location.

Another MSO option is to model immigration by periodically taking two particles in different swarms and exchanging them. One particle effectively immigrates into the current swarm and another particle emigrates out of the swarm. The demo program contains this option. The key method is:

```
private void Immigration(int i, int j)
{
    // Swap particle j in swarm i
    // with a random particle in a random swarm
    int otheri = ran.Next(0, swarms.Length);
    int otherj = ran.Next(0, swarms[0].particles.Length);
    Particle tmp = swarms[i].particles[j];
    swarms[i].particles[j] = swarms[otheri].particles[otherj];
    swarms[otheri].particles[otherj] = tmp;
}
```

There are several fascinating ways to modify the basic MSO algorithm.

The method is simple and allows the undesirable possibility that a particle might be exchanged with itself. The immigration feature is called inside method Solve like so:

```
double immigrate = 0.005;

...
double q = ran.NextDouble();
if (q < immigrate)
    Immigration(i, j);
```

Wrapping Up

The explanation presented in this article and the accompanying code download should give you a solid foundation for experimenting with multi-swarm optimization. Compared to regular

Figure 7 Computing the New Velocity of a Particle

```
v(t+1) = 0.7 * [-1.0, -3.0] +
    (1.4 * 0.2) * [8.0, 10.0] - [12.0, 24.0] +
    (1.4 * 0.2) * [7.0, 9.0] - [12.0, 24.0] +
    (0.4 * 0.2) * [5.0, 6.0] - [12.0, 24.0]

= 0.7 * [-1.0, -3.0] +
    0.3 * [-4.0, -14.0] +
    0.3 * [-5.0, -15.0] +
    0.1 * [-7.0, -18.0]

= [-0.7, -2.1] +
    [-1.2, -4.2] +
    [-1.5, -4.5] +
    [-0.7, -1.8]

= [-4.1, -12.6]
```

Figure 8 The First Lines of the Solve Method

```
public void Solve(int maxLoop)
{
    // Assign values to ct, w, c1, c2, c3
    double death = 0.005; // Prob of death
    while (ct < maxLoop)
    {
        ++ct;
        for (int i = 0; i < swarms.Length; ++i) {
            for (int j = 0; j < swarms[i].particles.Length; ++j) {
                double p = ran.NextDouble();
                if (p < death)
                    swarms[i].particles[j] = new Particle(dim, minX, maxX);
                for (int k = 0; k < dim; ++k) {
                    ...
                }
            }
        }
    }
}
```

particle swarm optimization, multi-swarm optimization is only slightly more complex, and tends to produce better-quality results, though it is slower. My experience with MSO suggests that it tends to handle difficult optimization problems—those with many local minima—better than regular particle swarm optimization.

MSO is a general-purpose numerical optimization meta-heuristic and is typically used as part of a larger machine-learning scenario in order to find a set of weights that minimize some sort of error function. For example, MSO can be used to find the best set of weights and bias values for a neural network by minimizing classification errors on a set of training data. There are many alternatives to MSO, including evolutionary optimization algorithms (also called real-valued genetic algorithms), bacterial foraging optimization and amoeba method optimization.

Compared to most alternative general-purpose numerical optimization approaches, in my opinion MSO is simpler to implement and easier to customize. A disadvantage of MSO compared to some alternatives is that in general there are few guidelines for selecting the value of MSO-free parameters, including the inertia weight constants and the cognitive, social and global weight constants. That said, however, I'm a big fan of MSO and it has performed extremely well for me on occasions when I needed to solve a numerical optimization problem in my software systems. ■

DR. JAMES McCaffrey works for Microsoft at the Redmond, Wash., campus. He has worked on several Microsoft products including Internet Explorer and MSN Search. He's the author of ".NET Test Automation Recipes" (Apress, 2006), and can be reached at jammc@microsoft.com.

THANKS to the following technical expert for reviewing this article: Kirk Olynyk (Microsoft)

facebook



Microsoft SharePoint 2010



Linked in



twitter

SEE THE WORLD AS A DATABASE

amazon web services

bing

amazon web services

Microsoft Excel 2010

Microsoft SQL Server

OData Open Data Protocol



ADO.NET ▪ JDBC ▪ ODBC ▪ SQL SSIS ▪ ODATA
MYSQL ▪ EXCEL ▪ POWERSHELL

Microsoft SQL Server

Linked in

SAP

OData Open Data Protocol

Salesforce



facebook



Microsoft SharePoint 2010

amazon web services



Microsoft Visual Studio Java ODBC Microsoft SQL Server Microsoft Excel Microsoft BizTalk MySQL OData

Work With Relational Data, Not Complex APIs or Services

Whether you are a developer using ADO.NET, JDBC, OData, or MySQL, or a systems integrator working with SQL Server or Biztalk, or even an information worker familiar with ODBC or Excel – our products give you bi-directional access to live data through easy-to-use technologies that you are already familiar with. If you can connect to a database, then you will already know how to connect to Salesforce, SAP, SharePoint, Dynamics CRM, Google Apps, QuickBooks, and much more!



Give RSSBus a try today and see what mean:

visit us online at www.rssbus.com to learn more or download a free trial.

rssbus

INTEGRATION YOUR WAY



Exploring NSpec

Readers of my column will know that one of the mechanisms I like to use to explore or demonstrate a new technology (library or otherwise) is what a colleague described once as an “exploration test”: a suite of unit tests designed not to test the technology, but explore it. It allows me—the developer/user—to validate some assumptions I’m making about the technology. More important, it gives me a regression suite to run when a new version of that technology comes out, so that if the new version makes some kind of critical breaking change, I can find out about it right away, and not after I’ve made the upgrade (and had to discover the breaking change at run time, surrounded in the context of my program, rather than the much lighter-weight context of a test suite).

On the subject of testing, however, there’s been some angst and consternation within the community over different “modes” of testing: unit testing versus Test-Driven Development (TDD) versus Behavior-Driven Development (BDD), usually accompanied with lots of shouting and screaming. Because of the religious nature of such debates, I’ve tended to avoid that subject matter, but recently (as part of doing some research for this column, in fact), I’ve discovered a testing tool—NSpec—that comes from the BDD crowd. Regardless of how much you like or dislike the BDD approach to testing, NSpec is worth exploring.

Getting Started

As with so many of the other Microsoft .NET Framework packages these days, getting started with NSpec is just a NuGet “Install-Package”

Figure 1 A Test with Two Unequal Strings

```

PM> NSpecRunner.exe .\Calculator\bin\debug\Calculator.dll

my first spec
  given the world has not come to an end
  Thunderhorse should be Thunderhorse - FAILED - Expected string
  length 12 but was 6. Strings differ at index 1.,
  Expected: "Thunderhorse", But was: "Hello ", -----^
**** FAILURES ****

nspec. my first spec. given the world has not come to an end.
Thunderhorse should be Thunderhorse.
Expected string length 12 but was 6. Strings differ at index 1.,
Expected: "Thunderhorse", But was: "Hello ", -----^
at Calculator.my_first_spec.<Thunderhorse_should_be_Thunderhorse>b__2() in
c:\Projects\Publications\Articles\MSDN\WorkingProg\NSpec\NSpec\
Calculator\
Class1.cs:line 19

2 Examples, 1 Failed, 0 Pending
PM>

```

command away. As a stalking horse for it, I begin by creating a Class Library solution (in C#, because it won the rock-paper-scissors-lizard-Spock contest that I had between it, F# and Visual Basic for rights to be the code to test) to build a Reverse Polish Notation (RPN) calculator class that I’ll call “NiftyCalc” for lack of anything more original. NiftyCalc is going to be a bit of a deviant from traditional RPN calculators, however, mostly so I can have something a little more complex to test.

There’s been some angst and consternation within the community over different “modes” of testing.

I’ll begin with bare-bones basics. After creating a new solution and Class Library project in Visual Studio, I’m going to create a simple class that has a single, auto-generated property member, named Current, to reflect the current value held in the calculator’s memory:

```

namespace Calculator
{
    public class NiftyCalc
    {
        public NiftyCalc()
        {
            Current = 0;
        }

        public int Current { get; set; }
    }
}

```

It ain’t much, but it gives enough scaffolding to get NSpec installed so I can look at the beginnings of using it.

After doing the “Install-Package nspec” in the Package Manager Console, a couple of things will have appeared. First, NSpec inserts another file into the Class Library project called DebuggerShim.cs. I won’t go into much detail on that now, but it lets you run NSpec tests via TDD and ReSharper, along with providing for continuous integration. More important, NSpec hasn’t created a standalone project as a peer to the Class Library project, like a standard Microsoft Test or NUnit test scenario would—the intent is that NSpec tests (or, as the company prefers to call them, specifications) are living inside the same project as the code they’re specifying.

HTML5+jQuery

Any App - Any Browser - Any Platform - Any Device



IGNITEUITM
INFRAGISTICS JQUERY CONTROLS



Download Your **Free Trial!**
www.infragistics.com/igniteui-trial



Infragistics Sales US 800 231 8588 • Europe +44 (0) 800 298 9055 • India +91 80 4151 8042 • APAC +61 3 9982 4545

Copyright 1996-2013 Infragistics, Inc. All rights reserved. Infragistics and NetAdvantage are registered trademarks of Infragistics, Inc.
The Infragistics logo is a trademark of Infragistics, Inc. All other trademarks or registered trademarks are the respective property of their owners.

Which means, then, that to create a test to test this, you can just create a class that extends the base class “nspec”:

```
using NSpec;

namespace Calculator
{
    class my_first_spec : nspec
    {
        void given_the_world_has_not_come_to_an_end()
        {
            it["Thunderhorse should be Thunderhorse"] =
                () => "Thunderhorse".should_be("Thunderhorse ");
        }
    }

    // ...
}

```

This clearly isn't a traditional unit test: it lacks the custom attribute tags; the naming convention has all those underscores in it; there's some kind of string-to-lambda dictionary at work here (“it”); and everything is essentially private/internal. To further make things significantly different, NSpec installed a new test-runner tool into the PATH inside the Package Manager Console, and that's where

Figure 2 Testing NiftyCalc Functionality

```
using NSpec;

namespace Calculator
{
    class nifty_calc : nspec
    {
        void calculator_basics()
        {
            it["New NiftyCalc should have a Current of 0"] =
                () => new NiftyCalc().Current.should_be(0);
            it["Pushing 5 should show Current as 5"] =
                () =>
                {
                    NiftyCalc nc = new NiftyCalc();
                    nc.Push(5);
                    nc.Current.should_be(5);
                };
            it["Push, push, pop, should have Current set to first pushed value"] =
                () =>
                {
                    NiftyCalc nc = new NiftyCalc();
                    nc.Push(5); nc.Push(7); nc.Pop();
                    nc.Current.should_be(5);
                };
        }
    }

    public class NiftyCalc
    {
        private Stack<int> theStack = new Stack<int>();

        public NiftyCalc()
        {
            theStack.Push(0);
        }

        public int Current
        {
            get { return theStack.Peek(); }
        }

        public void Push(int value)
        {
            theStack.Push(value);
        }

        public int Pop()
        {
            return theStack.Pop();
        }
    }
}

```

you turn in order to execute the tests. In the Package Manager Console (which, remember, is basically a Windows PowerShell console running inside Visual Studio), run NSpecRunner.exe with the full path to the compiled class library:

```
PM> install-package nspec
Successfully installed 'nspec 0.9.65'.
Successfully added 'nspec 0.9.65' to Calculator.

PM> NSpecRunner.exe .\Calculator\bin\debug\Calculator.dll

my first spec
  given the world has not come to an end
    Thunderhorse should be Thunderhorse

1 Examples, 0 Failed, 0 Pending
PM>

```

Now the odd naming conventions become clear: NSpec is using reflection to discover the classes and methods to execute as part of the test suite, and it uses the underscores in the names as spaces when printing the output. The intent here is to get to more natural-language-like syntax, so that I can write in “My first spec” that there's a specification, “Given the world has not come to an end, Thunderhorse should be Thunderhorse”; so long as the lambda expression associated with that last part of the spec yields true, everything is good. Should the spec/test be changed slightly such that the two strings aren't equal, NSpecRunner will give a different output (just as any test runner would), as shown in Figure 1.

As might be inferred, the actual “test” part of the spec is the should_be extension method that NSpec introduces onto anything that derives from System.Object. Further, there are a number of similarly named overloads on the idea, such that if you want to make sure 2+2 is greater than 3, you can use the should_be_greater_than method instead. Another half-dozen or so overloads of similar vein are available; anyone familiar with any unit-test framework should be comfortable with these.

One drawback to NSpec, at least the version (0.95) I tested, is that NSpec isn't deeply integrated with Visual Studio, so that you can't just hit F5 and build/run the projects; instead, you have to explicitly build the library, then find the Package Manager Console and up-arrow to the NSpecRunner line to kick off the tests. The Web site NSpec.org has a solution to this, called specwatchr, which is a Ruby script that keeps an eye on the files in the directory and kicks off NSpecRunner whenever a source file is saved. Although I don't mind it, some developers and some shops will turn up their nose at installing the Ruby platform just to have this functionality, useful as it may be. Personally, I think whatever distaste you might have for installing Ruby on your machine is vastly offset by having a tool that automatically runs tests like this, particularly as it has proven to be a highly useful technique in other language and platform environments, with the .NET Framework as the lone contrarian on this point. I recommend taking the time to give it a whirl—in a virtual machine (VM) that you don't care about, if you must.

Testing NiftyCalc

Meanwhile, with some of the NSpec basics out of the way, I can turn to putting some tests around NiftyCalc and extending it. For starters, I want to make sure that when a NiftyCalc is created, its current contents start at 0; when I push a number onto the NiftyCalc stack, that number is what's displayed as Current; and if

I push a number, push another number and pop the stack, the first number is retained, as shown in **Figure 2**.

So far, so good. In fact, it's almost redundant to describe in prose the test I want to write, then show the code for the test, because the very syntax of the NSpec test registry (the "it" instance inside of the NSpec-derived class) really allows for as verbose or as terse a description as I'd like. This is, as mentioned earlier, by design, so that tests and specifications are easy to read and understand—which, considering how often test suites are being used as documentation these days, is a valuable thing.

NiftyCalc isn't trying to be a stack.

Exceptions

Of course, there's also the matter of some edge cases to consider: What should happen in the event a user tries to pop one more value off of the NiftyCalc than was ever pushed? It's a common problem with stacks, but NiftyCalc isn't trying to be a stack, but rather a functional calculator. Toward that end, both the Pop method and the Current property's get method should check to see if there's anything in the stack to retrieve, and if not, yield a value of zero.

However, should the business prefer that NiftyCalc toss an exception when popping an empty stack, the NSpec code can check for that, like so:

```
it["Should throw a NullRef if the NiftyCalc is null"] =
  expect<NullReferenceException>( () => {
    NiftyCalc nc = null;
    nc.Push(5);
  });
```

Again, this isn't that far away from what a unit-test framework (such as Microsoft Test Manager or NUnit) looks like.

Context

There's a fair amount of repetition with the preceding NSpec code, by the way; it would be nice if you could set up a "context" in which the test is run, so you don't have to constantly allocate the NiftyCalc and push a few values in order to test. (This will be invaluable in the next set of tests, because I want to test the functionality of add, subtract and so on.) NSpec allows you to create such a context, as shown in **Figure 3**.

This test also shows how to make use of the "before" hook, to set up each individual test exercise before executing it—this gives you the chance to create the NiftyCalc instance and push two values to it before handing it into the test for execution.

Growing Usage

There's a whole slew more about NSpec—and the larger subject of BDD—beyond what's been discussed here, but this gets some of the basics in play. BDD enthusiasts will criticize the style and approach I've taken to writing my tests, but that's a stylistic and aesthetic debate I neither want to participate in nor try to capture in this article. NSpec itself is an interesting approach to testing—and more important, a component that's appearing in more and more open source projects. If you want to learn more about NSpec, Amir Rajan,

Figure 3 Setting Up a Test Context

```
using NSpec;

namespace Calculator
{
  class nifty_calc : nspec
  {
    void calculator_basics()
    {
      // ...
      void before_each()
      {
        Calc = new NiftyCalc(); //maybe move this line here
      }

      context["When NiftyCalc has a 2 and 3 pushed"] = () =>
      {
        before = () =>
        {
          calc.Push(3);
          calc.Push(2);
        };

        it["should Add to 5"] = () =>
        {
          calc.Add();
          calc.Current.should_be(5);
        };
        it["should Subtract to 1"] = () =>
        {
          calc.Subtract();
          calc.Current.should_be(1);
        };
      };
    }
    private NiftyCalc calc;
  }

  public class NiftyCalc
  {
    private Stack<int> theStack = new Stack<int>();

    public NiftyCalc() { }

    public int Current
    {
      get { return (theStack.Count != 0) ? theStack.Peek() : 0; }
    }

    public void Push(int value) {
      theStack.Push(value);
    }

    public int Pop() {
      return (theStack.Count != 0) ? theStack.Pop() : 0;
    }

    public void Add() { Push(Pop() + Pop()); }
    public void Subtract() { int top = Pop();
      int bot = Pop(); Push(bot - top); }
  }
}
```

codeveloper of the framework, has kindly volunteered to answer questions and provide guidance. You can reach him on Twitter at twitter.com/amirrajan or via his GitHub site at github.com/amirrajan.

Happy coding! ■

TED NEWARD is the principal of Neward & Associates LLC. He has written more than 100 articles and authored and coauthored a dozen books, including "Professional F# 2.0" (Wrox, 2010). He's an F# MVP and speaks at conferences around the world. He consults and mentors regularly—reach him at ted@tedneward.com if you're interested in having him come work with your team, or read his blog at blogs.tedneward.com.

THANKS to the following technical expert for reviewing this article:
Amir Rajan (*Improving Enterprises*)



Understanding Your Language Choices for Developing Modern Apps

Developing modern software using a single language just doesn't happen in 2013. Programming has evolved to a polyglot model with domain-specific languages (DSLs) for each tier of development. For example, a popular scenario is to use SQL and C# on the back end and HTML, JavaScript and CSS as the UI languages. As a developer, you want to be able to add skills to your skill set, but working in too many languages spreads your skills thin in each. You also want to put products out to market quickly, which often leaves little room for learning a new language and all its quirks, so optimally you want to build on top of skills you already have. When creating Windows Store and Windows Phone apps, you have a variety of choices in languages for every project scenario, and at least one will nicely complement your experience. In this article, I'll show you what languages go with different development scenarios. In addition, I'll look at the options you need to weigh and how factors such as data access or porting an app affect the language you need to use.

The Landscape of Modern App Languages

Here are the available language sets for Windows Store app development:

- HTML, JavaScript and CSS
- XAML and C#, or XAML and Visual Basic .NET
- XAML and C++, or DirectX and C++

If you develop on the Microsoft stack, at least some of these languages should be familiar and, of course, Web developers know HTML and friends quite well. It makes sense that many of the same language sets are available for Windows Phone development: XAML with C#, Visual Basic .NET, and C++.

What about HTML, JavaScript and CSS for Windows Phone? At the time of this writing, the only way you can write HTML in a Windows Phone app is by using a WebView control, a technique normally seen in hybrid apps (these are half-Web, half-native app mutants).

If you're creating a cross-platform native app (not a Web site) outside of the Microsoft ecosystem, the language landscape changes to these two options:

- Java for Android
- Objective-C for iOS (technically there are more options, but I'll discuss only Objective-C here)

Fortunately, Xamarin has come to the rescue for .NET and Windows developers who want to write cross-platform apps. There's no need to learn Java (although it's quite similar to C#) or Objective-C, the language of iOS development (more on this later).

Now that you have a solid idea of what languages are available to use, I'll look at how easy or difficult each one is to learn.

The Language Learning Curve

The path of least resistance is often the best path to take, and it's certainly the fastest. The languages you and your team are familiar with are an important consideration when moving into new territory. Not everyone seamlessly transitions from one language to another, particularly with languages that behave completely different. It's more difficult for some junior developers because they usually lack experience with different language paradigms and therefore more frequently make mistakes or are unaware of common pitfalls. Even seasoned developers who move from a compiled language such as C# to interpreted JavaScript can experience a lot of pain because they don't realize tiny quirks of the language such as the *this* keyword behaving differently, that hoisting occurs, or the myriad other JavaScript oddities.

Fortunately, Web developers continue to enjoy working with the same HTML, CSS and JavaScript as they would in any client Web project, but the difference when writing for Windows 8 is the addition of the Windows Library for JavaScript (WinJS). WinJS makes it easy to write native apps on Windows 8 with Web languages. You continue to use the same open standard APIs as in Web development, but you also work with Windows-specific libraries to access the native experience.

Developers who are accustomed to Web languages will find that the trick to HTML controls in Windows Store apps is that they're simply elements marked with `data-win*` attributes. Using these attributes allows the core WinJS code and some CSS to transform those elements into beautiful UI widgets that make a rich native experience. The WinJS code that renders the controls contains CSS rules that apply the look and feel of the Windows 8 modern UI. The core WinJS code also creates the more complex controls such as date pickers and grids at run time, and can add data-binding HTML elements to JavaScript arrays or objects.

XAML developers from both the Windows Presentation Foundation (WPF) and Silverlight camps continue to write the same declarative XAML for the UI layout alongside C#, Visual Basic .NET or C++ as its compiled companion. As you might expect, in XAML there are new libraries and namespaces dedicated to the Windows 8 modern experience.

There's a great amount of parity among the declarative languages—that is, any “ML” language—concerning the amount and quality of UI controls. In general, if you find a control in XAML, it likely exists in HTML or the WinJS library and vice versa; however, there are one or two differences between the two.



YOUR .NET Resources



Visual Studio[®]
MAGAZINE

Visual Studio **LIVE!**
EXPERT SOLUTIONS FOR .NET DEVELOPERS

ONLINE | NEWSLETTERS | PRINT | CONFERENCES

XAML developers will hardly notice a change except for some namespaces and other trivial changes. Expression Blend is still a great tool to do UI design in XAML, while Visual Studio is best for the compiled language behind the scenes. A widespread third-party ecosystem exists around XAML UI controls, such as grids for XAML on the WPF platform, and many of these controls work the same way or similarly between Windows Store and Windows Phone apps.

Finally, if you're a Windows Forms developer, it's time to learn XAML, as you practically have no other choice. There's no comparable UI language that you can use, because Windows Forms doesn't have one—it's all C# or Visual Basic .NET behind a design palette. While you can carry your general .NET and language skills over, you must learn the DSL of the UI: XAML. You could learn HTML, but that would require the addition of both CSS and JavaScript, so you'd then have three languages to learn, and that's no small undertaking.

Data Access Affects Language Choice

How you access data varies—sometimes greatly—between languages, and the kind of data you're dealing with determines how you access that data, as well. To start, some techniques are available only to one set of technologies or another, as is the case with Web Storage (also known as DOM Storage, which you can learn more about in my blog post, "Managing Data in Web Applications with HTML5 Web Storage," at bit.ly/lml0U1) and IndexedDB, both of which you may only use with browser-based languages such as JavaScript.

Expression Blend is still a great tool to do UI design in XAML.

Windows Azure Mobile services (which you can learn more about in another blog post, "Use Windows Azure Mobile Services to Power Windows Store and Windows Phone Apps," at bit.ly/15nh08d) and the Windows Azure platform offer several native APIs for all the popular platforms as well as REST APIs that are inherently cross-platform. Because REST is an open standard, you can access its data from any language, anywhere, at any time, via a consistent interface. REST interfaces make it easy to develop across platforms because—despite language syntax and quirks—the code has a general consistency, or pattern. Public APIs, such as those provided by Twitter, Facebook, weather services and so on, usually return JSON or XML (or both) that you can consume via REST or XMLHttpRequest (XHR).

SQLite is a local database option available across all platforms and languages. SQLite is available as a NuGet package for Windows Store and Windows Phone apps in Visual Studio 2012. You can also access it through a Xamarin toolset and use its libraries to access SQLite databases on iOS and Android. SQLite is easier to use with XAML apps than HTML apps.

File access and file I/O on the Windows platform are at par level between Windows Store XAML and HTML apps, so the language doesn't really matter when your app requires reading and writing to the file system. Although normal HTML and JavaScript

in the browser have many restrictions on file I/O operations and accessing system resources, the Windows Store app development model allows more than sufficient access to the file system while maintaining security.

For complete details on data-access technologies for Windows Store and Windows Phone apps, see my March 2013 column, "Data Access and Storage Options in Windows Store Apps," at msdn.microsoft.com/magazine/jj991982.

Cross-Platform Considerations

Objective-C is syntactically and operationally different from the other languages, as Java and C# behave similarly because they're managed languages. However, if you're writing cross-platform apps, you aren't doomed to separate codebases such as Objective-C for iOS, Java for Android and C# for Windows platforms, because the aforementioned Xamarin tools make it easy to write in C# and generate Java and Objective-C. Xamarin provides a native code-generation toolset for cross-platform device development (that is, mobile and tablet devices), and it's the lowest common denominator for doing cross-platform native development. As a result, developers from just about any platform can easily move their skills to the Windows platform or cross-platform via Xamarin and C#. Developers from the iOS and Android platforms would do well to pick C# as their language of choice if they want to publish on the Windows platform as well as the others. Note, however, that some developers in Redmond disagree, advocating that it's more convenient to develop libraries in C++ and C because they can be accessed by iOS (directly) and Windows apps (via Windows Runtime, or WinRT, components). They point out that this approach is "free" (most Xamarin tools cost money) and more performant than third-party tools. I favor just creating and sharing Xamarin components, as time to market is more important than the negligible performance benefits—and I don't particularly like using C++.

Porting from Legacy Codebases Affects Language Choice

Not all apps are greenfield apps (that is, new or rewritten). Most, in fact, are brownfield—or legacy programs—especially if the code you're writing is for the enterprise. Many apps are really parts of a larger ecosystem of software made of multiple apps or multiple UIs that interact with business logic, service layers, public APIs or libraries.

Migrating Web apps in particular can be tricky, as you often need to keep the content of an existing Web site intact yet integrate that same content into the client app and make it feel like a native experience. This is because business requirements frequently dictate that the original Web site or architecture must remain unchanged. This can cause the look and feel of the app to be different from its host OS, making usability difficult, and it can block the user's natural workflow. Fortunately, if you're porting an existing Web site or app, you often have the choice to integrate instead. If this is the case, then the Apache Cordova framework (formerly PhoneGap) can help. Cordova enables you to create hybrid, Web-native apps by integrating Web content and native UI components and publishing across platforms. Conversely, integrating Web content with a native WinJS Windows Store app isn't difficult to do using WinJS, and the

experience is usually smoother for the user. Native apps enjoy a closer integration experience with the OS than client Web apps do, because native apps have access to modern hardware resources such as the camera, microphone, device settings, accelerometers, geolocation and so on. In addition to this, native apps have a distinct lifecycle and contain state, whereas Web apps are lightweight and stateless.

The Purpose of the App Matters

The type and purpose of your app determine the language you use, to an extent. High-intensity games and graphics apps lend themselves to languages such as C++ and DirectX (now part of the Windows SDK) because you can get closer to the machine and gain performance and better device control. DirectX is a complete collection of close-to-the-metal APIs you can access via C++ to create stunning games and apps. The downside of C++/DirectX is complex code that makes it easy to get yourself into trouble. C++ isn't the only option for games, either. JavaScript—yes, JavaScript—apps are top performers, especially for canvas games. The Windows Store app execution container offloads script processing to the GPU so JavaScript can run in parallel yet independent of the layout done by the Trident layout engine. This notion of offloading makes JavaScript a really fast language that can perform just as well as native code. If performance is a big deal for you but you don't want to learn C++, using the HTML canvas means you might not have to.

If you write business and productivity apps, you usually have your choice of any language that suits you, because things such as performance and UI controls are so similar. These apps tend to be common forms-over-data apps. The current set of technologies is what normally drives language and technology choices for these kinds of apps.

What About F#?

Wondering what happened to F#? While F# isn't one of the basic templates for Windows Store and Windows Phone app development, you can reference and consume components created in F#. It's also a great language for processing analytics and running code on the back end—and all apps need a back end.

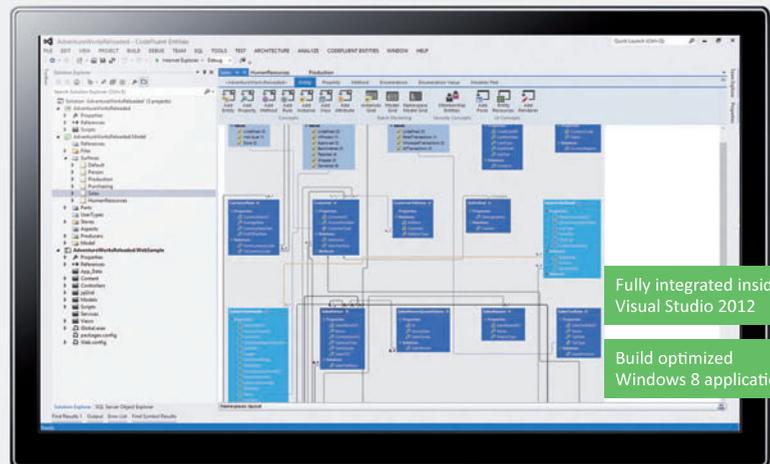
Web developers often use a mix of CSS for styling HTML and JavaScript that together build modern UIs. Native developers also have a set of concurrent languages for the same, so it's up to you to consider all the factors that might affect your app when choosing a language. ■

RACHEL APPEL is a consultant, author, mentor and former Microsoft employee with more than 20 years of experience in the IT industry. She speaks at top industry conferences such as Visual Studio Live!, DevConnections, MIX and more. Her expertise lies within developing solutions that align business and technology focusing on the Microsoft dev stack and open Web. For more about Appel, visit her Web site at rachelappel.com.

THANKS to the following Microsoft technical experts for reviewing this article: John Kennedy and Eric Schmidt

Save your time!

Stop writing repetitive code and focus on what matters



Fully integrated inside Visual Studio 2012

Build optimized Windows 8 applications

Generate rock-solid foundations for your .NET applications



Benefit from out-of-the-box advanced features

“ A remarkable product that adds features where all the ‘junior’ equivalents fall short. Things like hassle-free schema updates, up-casting, enum & null management, security, full data-binding including grids with pagination, performance, interface support, custom stored procedures within a wide range of architectures.

Basically all the ‘add-ons’ you discover you need when you start developing a real world app based on any code generator. ”

Boris Bosnjak, Developer, Dreamquest, Canada

Get a license worth \$949 for free until September 30th

Go to www.softfluent.com/forms/msdn-q3-special-offer

Tools for developers, by developers.
More information at www.softfluent.com
Contact us at info@softfluent.com





Direct2D Geometries and Their Manipulations

High school geometry comes in two distinct flavors: Euclidean geometry is oriented around constructions, theorems and proofs, while analytic geometry describes geometric figures numerically using points on a coordinate system, often called the Cartesian coordinate system in honor of the pioneer of analytic geometry, René Descartes.

It's analytic geometry that forms the basis of the whole vector wing of computer graphics, so it's right and proper that an interface named ID2D1Geometry (and the six interfaces that derive from it) pretty much sits in the center of Direct2D vector graphics.

In the previous installment of this column (msdn.microsoft.com/magazine/dn342879), I discussed how to use the ID2D1PathGeometry to render lines in a finger-painting application that runs under Windows 8. Now I'd like to step back and explore geometries in broader detail, and in particular investigate some of the intriguing methods defined by ID2D1Geometry for manipulating geometries to make different geometries.

Even if you're familiar with using geometries in the Windows Runtime (WinRT), ID2D1Geometry provides facilities that aren't exposed in the WinRT Geometry class.

Overview

Geometries are basically collections of coordinate points. These coordinate points aren't tied to any particular device, so geometries are inherently device-independent objects. For that reason, you create geometries of various sorts by calling methods on the ID2D1Factory object, which is the device-independent Direct2D factory. The six geometry-creation methods are shown in **Figure 1**, along with the interface type of the objects they create.

With the exception of CreatePathGeometry, these methods create immutable objects: All the information necessary to create the object is passed to the creation method, and you can't change anything about the geometry after it has been created.

The only reason the ID2D1PathGeometry object is different is because CreatePathGeometry returns an object that's essentially empty. I'll describe how you fill it up shortly.

CreateTransformedGeometry accepts an existing ID2D1Geometry object and an affine transform matrix. The resultant geometry is translated, scaled, rotated or skewed by that matrix. The CreateGeometryGroup method accepts an array of ID2D1Geometry

Figure 1 The Six Geometry Methods and Interfaces

ID2D1Factory Method	Created Object (Derived from ID2D1Geometry)
CreateRectangleGeometry	ID2D1RectangleGeometry
CreateRoundedRectangleGeometry	ID2D1RoundedRectangleGeometry
CreateEllipseGeometry	ID2D1EllipseGeometry
CreatePathGeometry	ID2D1PathGeometry
CreateTransformedGeometry	ID2D1TransformedGeometry
CreateGeometryGroup	ID2D1GeometryGroup

objects and creates a geometry that is the composite of all the individual geometries.

If you use the Visual Studio Direct2D (XAML) project template to create a Windows Store application that accesses DirectX, you'll generally create geometry objects in the CreateDeviceIndependent-Resources override of the rendering class and use them throughout the lifetime of the program, in particular during the Render override.

Even if you're familiar with using geometries in the Windows Runtime, ID2D1Geometry provides facilities that aren't exposed in the WinRT Geometry class.

The ID2D1RenderTarget interface (from which interfaces such as ID2D1DeviceContext derive) defines two methods for rendering geometries on a drawing surface: DrawGeometry and FillGeometry.

The DrawGeometry method draws the lines and curves of the geometry with a specified brush, stroke thickness, and stroke style, allowing solid, dotted, dashed, or custom dash-patterned lines. The FillGeometry method fills enclosed areas of a geometry with a brush and an optional opacity mask. You can also use geometries for clipping, which involves calling PushLayer on the render target with a D2D1_LAYER_PARAMETERS structure that includes the geometry.

Code download available at archive.msdn.microsoft.com/mag201309DXF.

Figure 2 A Method to Build a Five-Pointed Star Geometry

```
HRESULT GeometryVarietiesRenderer::CreateFivePointedStar(
    float radius, ID2D1PathGeometry** ppPathGeometry)
{
    if (ppPathGeometry == nullptr)
        return E_POINTER;

    HRESULT hr = m_d2dFactory->CreatePathGeometry(ppPathGeometry);
    ComPtr<ID2D1GeometrySink> geometrySink;

    if (SUCCEEDED(hr))
    {
        hr = (*ppPathGeometry)->Open(&geometrySink);
    }

    if (SUCCEEDED(hr))
    {
        geometrySink->BeginFigure(Point2F(0, -radius), D2D1_FIGURE_BEGIN_FILLED);

        for (float angle = 2 * XM_2PI / 5; angle < 2 * XM_2PI; angle += 2 * XM_2PI / 5)
        {
            float sin, cos;
            D2D1SinCos(angle, &sin, &cos);
            geometrySink->AddLine(Point2F(radius * sin, -radius * cos));
        }

        geometrySink->EndFigure(D2D1_FIGURE_END_CLOSED);
        hr = geometrySink->Close();
    }

    return hr;
}
```

Figure 3 Much of the CreateDeviceIndependentResources Override

```
void GeometryVarietiesRenderer::CreateDeviceIndependentResources()
{
    DirectXBase::CreateDeviceIndependentResources();

    // Create square-wave geometry
    HRESULT hr = m_d2dFactory->CreatePathGeometry(&m_squareWaveGeometry);
    ComPtr<ID2D1GeometrySink> geometrySink;
    hr = m_squareWaveGeometry->Open(&geometrySink);

    geometrySink->BeginFigure(Point2F(-250, 50), D2D1_FIGURE_BEGIN_HOLLOW);
    geometrySink->AddLine(Point2F(-250, -50));
    geometrySink->AddLine(Point2F(-150, -50));
    geometrySink->AddLine(Point2F(-150, 50));
    geometrySink->AddLine(Point2F(-50, 50));
    geometrySink->AddLine(Point2F(-50, -50));
    geometrySink->AddLine(Point2F(50, -50));
    geometrySink->AddLine(Point2F(50, 50));
    geometrySink->AddLine(Point2F(150, 50));
    geometrySink->AddLine(Point2F(150, -50));
    geometrySink->AddLine(Point2F(250, -50));
    geometrySink->AddLine(Point2F(250, 50));
    geometrySink->EndFigure(D2D1_FIGURE_END_OPEN);

    hr = geometrySink->Close();

    // Create star geometry and translate it
    ComPtr<ID2D1PathGeometry> starGeometry;
    hr = CreateFivePointedStar(150, &starGeometry);
    hr = m_d2dFactory->CreateTransformedGeometry(starGeometry.Get(),
                                                Matrix3x2F::Translation(0, -200),
                                                &m_starGeometry);

    // Create infinity geometry and translate it
    ComPtr<ID2D1PathGeometry> infinityGeometry;
    hr = CreateInfinitySign(100, &infinityGeometry);
    hr = m_d2dFactory->CreateTransformedGeometry(infinityGeometry.Get(),
                                                Matrix3x2F::Translation(0, 200),
                                                &m_infinityGeometry);

    // Create geometry group
    CreateGeometryGroup();
    ...
}
```

Sometimes you need to animate a geometry. The most efficient approach is applying a matrix transform to the render target before rendering the geometry. However, if this isn't adequate, you'll need to re-create the geometry, probably during the Update method in the rendering class. (Or, you might want to create a bunch of geometries at the outset and save them.) Although re-creating geometries increases the rendering overhead, it's certainly necessary sometimes. But, for the most part, if you don't need to re-create geometries, don't do so.

The Geometry Sink

The ID2D1PathGeometry object is a collection of straight lines and curves, specifically cubic and quadratic Bezier curves and arcs, which are curves on the circumference of an ellipse. You can control whether these lines and curves are connected, and whether they define enclosed areas.

Filling a path geometry with lines and curves involves the use of an object of type ID2D1GeometrySink. This particular sink is not for washing geometries! Think of it as a receptacle—a destination for lines and curves that are then retained by the path geometry.

Here's how to build a path geometry:

1. Create an ID2D1PathGeometry object by calling CreatePathGeometry on an ID2D1Factory object.
2. Call Open on the ID2D1PathGeometry to obtain a new ID2D1GeometrySink object.
3. Call methods on the ID2D1GeometrySink to add lines and curves to the path geometry.
4. Call Close on the ID2D1GeometrySink.

The ID2D1PathGeometry isn't usable until Close has been called on the ID2D1GeometrySink object. After Close is called, the ID2D1PathGeometry is immutable: Its contents can't be altered in any way, and you can't call Open again. The geometry sink no longer has a purpose, and you can get rid of it.

The third item in the list usually involves the most extensive code. A path geometry is a collection of figures; each figure is a series of connected lines and curves, called segments. When calling functions on the ID2D1GeometrySink, you begin with an optional call to SetFillMode to indicate the algorithm used for filling enclosed areas. Then, for each series of connected lines and curves in the path geometry:

1. Call BeginFigure, indicating the first point and whether enclosed areas will be filled.
2. Call methods beginning with the word Add to add connected lines, Bezier curves and arcs to the figure.
3. Call EndFigure, indicating whether the last point should be automatically connected to the first point with a straight line.

When you look at the documentation of ID2D1GeometrySink, take note that the interface derives from ID2D1SimplifiedGeometrySink, which actually defines the most important methods. More on this differentiation shortly.

Because the contents of an ID2D1PathGeometry are immutable once you've defined its path, if you need to alter an ID2D1PathGeometry you'll have to re-create it and go through the path definition process again. However, if you're just adding additional figures to the beginning or end of an existing path geometry, then a shortcut

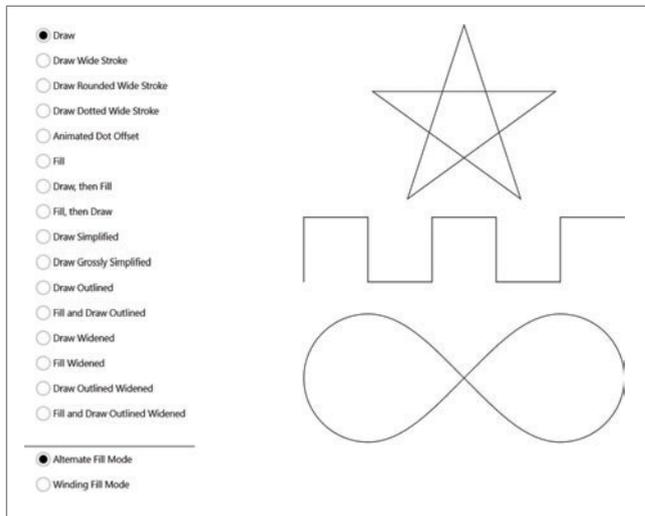


Figure 4 The Startup Screen of GeometryExperimentation

is available: You can create a new path geometry, add some figures to it, and then transfer the contents of the existing path geometry into the new path geometry by calling the Stream function of the existing path geometry with the new ID2D1GeometrySink. You can then add additional figures before closing.

Drawing and Filling

Now I'm ready to show you some code. The downloadable Geometry-Experimentation project was created in Visual Studio 2012 using the Windows Store Direct2D (XAML) template. I renamed the SimpleTextRenderer class to GeometryVarietiesRenderer, and I removed all the code and markup associated with the sample text rendering.

If you need to alter an
ID2D1PathGeometry you'll have
to re-create it and go through the
path definition process again.

In the XAML file, I defined a bunch of radio buttons for various geometry-rendering techniques. Each radio button is associated with a member of a RenderingOption enumeration I defined. A big switch and case statement in the Render override uses members of this RenderingOption enumeration to govern what code is executed.

All the geometry creation occurs during the CreateDeviceIndependentResources override. One geometry that the program creates is a five-pointed star. The somewhat-generalized method that builds this geometry is shown in **Figure 2**. It consists of one figure with four line segments, but the last point is automatically connected to the first point.

Figure 3 shows much of the CreateDeviceIndependentResources method. (To keep the listing simple, I removed the handling of errant HRESULT values.) The method begins by creating a

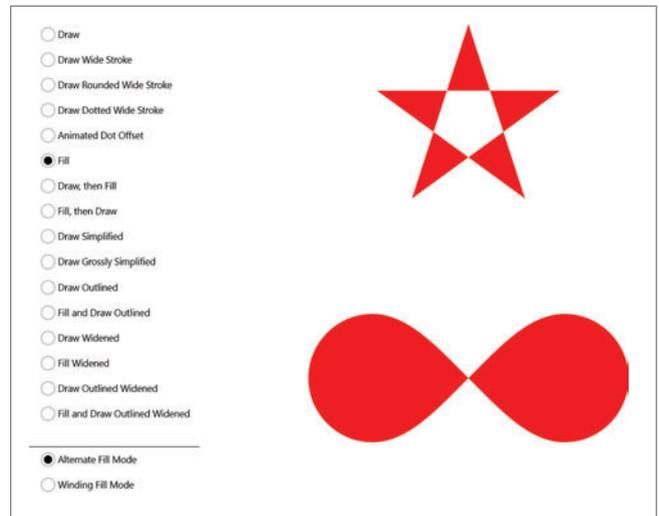


Figure 5 Filling Geometries

geometry that resembles a square-wave, a call to create the five-pointed star, and a call to another method to create an infinity sign. Two of these geometries are transformed, and all three are combined in the CreateGeometryGroup method (called at the bottom of **Figure 3**) into a member named m_geometryGroup.

The CreateDeviceIndependentResources method also creates two stroke styles with rounded ends and joins. One is solid and the other is dotted.

The CreateDeviceDependentResources method creates two brushes: black for drawing and red for filling.

When the program starts up, the first radio button is checked, and DrawGeometry is called:

```
m_d2dContext->DrawGeometry(m_geometryGroup.Get(),
                           m_blackBrush.Get());
```

The result is shown in **Figure 4**, looking oddly like the logo of some weird graphics cult.

Because the program combined the three separate geometries to simplify the rendering, they'll all be rendered with the same brush. In a real program, you'd probably maintain a bunch of individual geometries and color them all differently.

The first few options demonstrate how the geometry can be drawn with a thicker stroke and a styled line, including a dotted line. (Throughout the program, stroke thicknesses of 1, 10 and 20 pixels are used, and should be easily distinguishable visually.)

When demonstrating path geometries and animation in XAML, I like to apply a XAML-based animation to the dash offset of a stroke style, causing dots to travel around the geometry. You can do something similar in DirectX (as the Animated Dot Offset option demonstrates), but you need to explicitly recreate the ID2D1StrokeStyle object during each screen refresh. This happens in the Update method.

```
Enclosed areas of the geometry can also be filled with a brush:
m_d2dContext->FillGeometry(m_geometryGroup.Get(),
                          m_redBrush.Get());
```

The result is shown in **Figure 5**. There are no enclosed areas in the square wave. The interior pentagon of the five-pointed star isn't filled because the filling mode is set to an algorithm known as Alternate. You can use the pair of radio buttons at the bottom left of **Figure 5** to select Winding Fill Mode to fill the

pentagon. The fill mode needs to be specified when creating the `ID2D1GeometryGroup`, so the `m_geometryGroup` object needs to be re-created when either of those two radio buttons are clicked. If the geometries overlap in the geometry group, then intersecting areas are filled also based on that fill mode.

Often you'll want to both draw and fill a geometry. Generally you'll want to call `FillGeometry` first and then `DrawGeometry` to keep the stroke fully visible.

Simplified and Simplifying Geometries

Suppose your application creates an `ID2D1PathGeometry` dynamically, perhaps from user input, and you'd like to "interrogate" the path geometry to extract all the figures and segments. Perhaps you'd like to save this information in a XAML format as a series of standard `PathGeometry`, `PathFigure`, `LineSegment` and `BezierSegment` tags.

The key here is that you can write your own class that implements the `ID2D1GeometrySink` interface, and pass an instance of that class to the `Stream` method.

At first, it doesn't appear as if this is possible. The `ID2D1PathGeometry` has `GetFigureCount` and `GetSegmentCount` methods, but no methods to actually extract those figures and segments.

But recall the `Stream` method. This method accepts an `ID2D1GeometrySink` and copies the contents of the path geometry into that sink. The key here is that you can write your own class that implements the `ID2D1GeometrySink` interface, and pass an instance of that class to the `Stream` method. Within this custom class, you can handle all the calls to `BeginFigure`, `AddLine` and so forth, and do whatever you want with them.

Of course, such a class isn't trivial. It would need implementations of all the methods in `ID2D1GeometrySink`, as well as `ID2D1SimplifiedGeometrySink` and `IUnknown`.

However, there's a way to make this job somewhat easier: The `ID2D1Geometry` interface defines a method named `Simplify` that converts any geometry object into a "simplified" geometry that contains only straight lines and cubic Bezier splines. This feat is

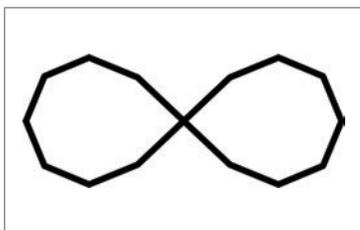


Figure 6 A Grossly Flattened Infinity Sign

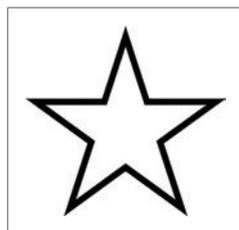


Figure 7 An Outlined Path Geometry

possible because quadratic Bezier splines and arcs can be approximated by cubic Bezier splines. This means your custom class need only implement the `ID2D1SimplifiedGeometrySink` and `IUnknown` methods. Simply pass an instance of this custom class to the `Simplify` method of any geometry.

You can also use `Simplify` to copy the simplified contents of a geometry to a new path geometry. Here's the code in `GeometryExperimentation` that does this (excluding the `HRESULT` checking):

```
m_d2dFactory->CreatePathGeometry(&m_simplifiedGeometry);
ComPtr<ID2D1GeometrySink> geometrySink;
m_simplifiedGeometry->Open(&geometrySink);
m_geometryGroup->Simplify(D2D1_GEOMETRY_SIMPLIFICATION_OPTION_CUBICS_AND_LINES,
    IdentityMatrix(), geometrySink.Get());
geometrySink->Close();
```

Notice the first argument to the `Simplify` method indicates that you want both cubic Beziers and straight lines in the simplified path geometry. You can restrict that to only lines, in which case the Bezier curves are approximated by a series of straight lines. This is a process called "flattening." If flattening is performed with a lot of precision you can't tell the difference visually, but you can also specify a flattening tolerance so the Bezier curve is *not* approximated very well. Here's some code in `GeometryExperimentation` that creates a "grossly simplified" geometry:

```
m_d2dFactory->CreatePathGeometry(&m_grosslySimplifiedGeometry);
m_grosslySimplifiedGeometry->Open(&geometrySink);
m_geometryGroup->Simplify(D2D1_GEOMETRY_SIMPLIFICATION_OPTION_LINES,
    IdentityMatrix(), 20, geometrySink.Get());
geometrySink->Close();
```

The star and square wave look the same, but the infinity sign is no longer quite so smooth, as shown in **Figure 6**. By default, the flattening tolerance is 0.25.

More Geometry Manipulations

The `ID2D1Geometry` interface defines three additional methods that are similar to `Simplify` in that they calculate a new geometry and write it into an `ID2D1SimplifiedGeometrySink`. I'll discuss `Outline` and `Widen` here, but not `CombineWithGeometry` (because it only does something interesting with multiple overlapping geometries, and I have none in this program).

As you've seen, path geometries can have intersecting segments. The infinity sign has an intersecting segment right in the center, and the five-pointed star has a bunch of intersecting segments. The `Outline` method defined by `ID2D1Geometry` creates a new path geometry based on an existing path geometry that eliminates those intersections but retains the same enclosed areas.

Here's the code in `GeometryExperimentation` that converts the geometry group into an outlined path geometry:

```
m_d2dFactory->CreatePathGeometry(&m_outlinedGeometry);
m_outlinedGeometry->Open(&geometrySink);
m_geometryGroup->Outline(IdentityMatrix(), geometrySink.Get());
geometrySink->Close();
```

Because this `m_outlinedGeometry` object defines the same filled areas as `m_geometryGroup`, it's different depending on whether `m_geometryGroup` was created using an `Alternate Fill Mode` or `Winding Fill Mode`.

The original geometry group has a total of three figures: one for the star, one for the square wave and one for the infinity sign. If this geometry group is created with the `Alternate Fill Mode`, the outlined geometry contains eight figures: Five for the five points of the star, one for the square wave and two for the infinity sign.

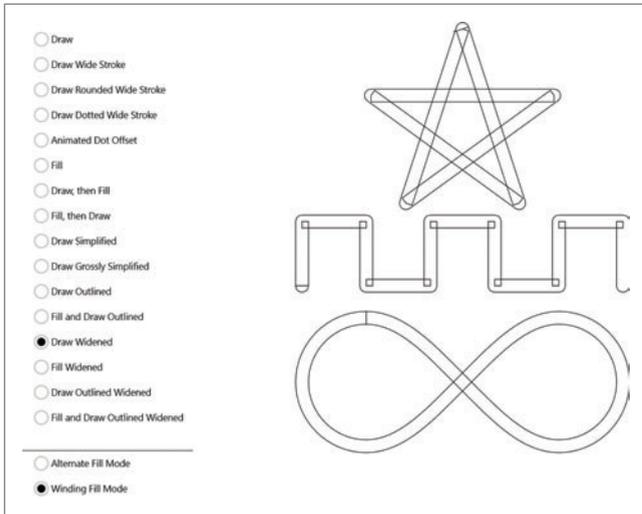


Figure 8 A Widened Path Geometry

But visually it seems the same. If the geometry group is created with the Winding Fill Mode, however, the outlined geometry has a total of four figures: the infinity sign has two figures just as with the Alternate Fill Mode, but because the entire interior of the star is filled, the star consists of just one figure, as shown in Figure 7.

Defining such a geometry from scratch would be mathematically rather difficult, but the Outline method makes it quite easy. Because the path geometry defined by Outline contains no intersecting segments, the fill mode has no effect.

If flattening is performed with a lot of precision you can't tell the difference visually.

I find the Widen method to be the most interesting of all. To understand what Widen does, consider a path geometry that contains just a single straight line between two points. When this geometry is drawn, it's stroked with a particular line thickness, so it's actually rendered as a filled rectangle. If the line style includes rounded ends, this rectangle is adorned with two filled semicircles.

The Widen method computes a path geometry that describes the outline of this rendered object. To do this, Widen requires arguments specifying the desired stroke thickness and stroke style, just like DrawGeometry. Here's the code in the GeometryExperimentation project:

```
m_d2dFactory->CreatePathGeometry(&m_widenedGeometry);
m_widenedGeometry->Open(&geometrySink);
m_geometryGroup->Widen(20, m_roundedStrokeStyle.Get(),
    IdentityMatrix(), geometrySink.Get());
geometrySink->Close();
```

Notice the 20-pixel stroke thickness. The program then draws this widened geometry using a one-pixel stroke thickness:

```
m_d2dContext->DrawGeometry(m_widenedGeometry.Get(),
    m_blackBrush.Get(), 1);
```

The result is shown in Figure 8. I find the artifacts created by this process extremely interesting. It's as if I'm somehow peering into the innards of the line-drawing algorithm.

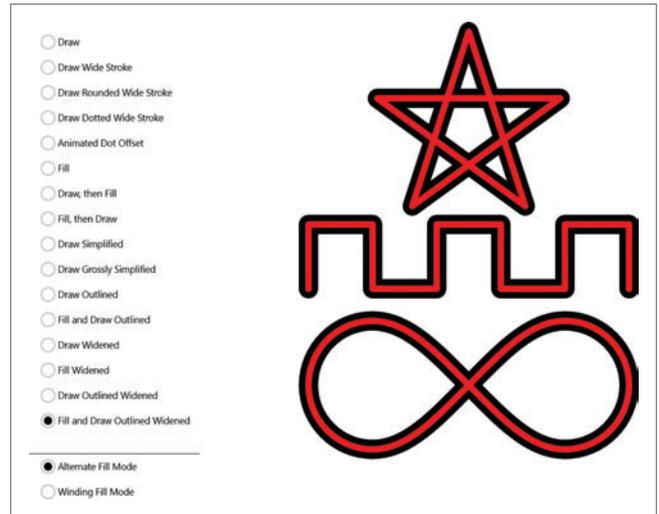


Figure 9 An Outlined Widened Path Geometry, Stroked and Filled

The GeometryExperimentation program also allows filling the widened path geometry:

```
m_d2dContext->FillGeometry(m_widenedGeometry.Get(),
    m_redBrush.Get());
```

Filling a path geometry that has been widened with a particular stroke width and stroke style is visually identical to stroking the original path with that width and style. The only visual difference between the Draw Rounded Wide Stroke and Fill Widened options in GeometryExperimentation is the color, because I use black for drawing and red for filling.

You might want to fill and draw a widened path geometry, but you'd probably prefer the internal artifacts are removed. Doing so is exceptionally easy. Simply apply the Outline method to the widened path geometry:

```
m_d2dFactory->CreatePathGeometry(&m_outlinedWidenedGeometry);
m_outlinedWidenedGeometry->Open(&geometrySink);
m_widenedGeometry->Outline(IdentityMatrix(), geometrySink.Get());
geometrySink->Close();
```

Now when you fill and draw the outlined widened path geometry, you get the image in Figure 9. It's free of artifacts and visually quite different from anything else rendered by this program, and different from anything I'd be brave enough to code from scratch.

Any Others?

There's another method that writes geometry data to an ID2D1SimplifiedGeometrySink, but you might have to go hunting for it. It's not among the Direct2D interfaces. It's in DirectWrite, and it's the GetGlyphRunOutline method of IDWriteFontFace. This powerful method generates path geometries from text character outlines, and it's simply too fun to ignore. Stay tuned. ■

CHARLES PETZOLD is a longtime contributor to MSDN Magazine and the author of "Programming Windows, 6th edition" (O'Reilly Media, 2012), a book about writing applications for Windows 8. His Web site is charlespetzold.com.

THANKS to the following technical experts for reviewing this article: Wessam Bahnassi (InFramez Technology), Worachai Chaoweeraprasit (Microsoft), Anthony Hodsdon (Microsoft) and Michael B. McLaughlin (Bob Taco Industries)

MSDN Magazine Online

MSDN Magazine  United States – English – Sign in

Home Topics Issues and Downloads Script Junkie Subscribe Submit an Article RSS msdn

THE MICROSOFT JOURNAL FOR DEVELOPERS

AUGUST 2013 VOL. 28 NO. 8

msdn magazine

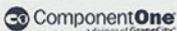


SCAN, VIEW, & ANNOTATE!
NET SDKs
Atalasoft
A Nokia Company
CAPTURE-ENABLE YOUR APPLICATIONS

Bringing RESTful Services to C++ Developers
Learn how to use the C++ REST SDK to build a simple Windows-based client application that uploads a file to Dropbox, along with a standard C++ class that supports OAuth.
Sridhar Poduri



DirectX Factor: Finger Painting with Direct2D Geometries
In a multi-touch environment like Windows 8, one kind of program every developer should know how to code is a finger painting app, which involves tracking individual fingers to draw lines on the screen. Charles Petzold explores how to do this using DirectX.
Charles Petzold

Studio Enterprise
Includes 10 Studios - 300+ controls
HTML5 · WinForms · ASP.NET · WPF · Silverlight
MVC · Windows Phone · WinRT XAML + more
ALL NEW 2013 v2!
Download 

Features

 **DirectX: Real-Time, Realistic Page Curling with DirectX, C++ and XAML**
Project Austin is a digital note-taking app for Windows 8 that's written in C++ and uses DirectX and XAML. Eric Brumer explores the geometry and programming that underlie pages that visually curl and uncurl like real paper.

Columns

Windows with C++
The Windows Runtime Application Model
Kenny Kerr explains how the best way to understand the Windows Runtime

MSDN Magazine Blog

MSDN Magazine June Issue Preview
Monday morning the June issue of MSDN Magazine will go live on our Web site. Here's what you can expect to find in the magazine. Windows 8 figures pr... [More...](#)
Friday, May 31

It's like *MSDN Magazine*—only better. In addition to all the great articles from the print edition, you get:

- Code Downloads
- The *MSDN Magazine* Blog
- Digital Magazine Downloads
- Searchable Content

All of this and more at msdn.microsoft.com/magazine

msdn
magazine



Teenagers

This is the first column I've written as the father of a teenager. Those of you who've been here know what it's like. For those who haven't yet, perhaps you can recall being on the other end of it.

When my daughter was born 13 years ago, I could never imagine this day. I was the World's Best Daddy when she was 5 and her sister was 3. But like all days, it rolled steadily nearer, one day at a time, and the next thing I knew, bang! My beautiful, scary-smart, happy toddler ("Daddy, look, an anthropomorphic snowman") is a beautiful, scary-smart, eye-rolling, grumpy teenager ("Daddy, you suck. And here's my cell phone bill").

I might take singer Don White's advice, and begin every sentence with the words, "I'm not trying to oppress you ..." (check him out on YouTube at bit.ly/18xLpQ7 and bit.ly/1c0gl49). As White quips of the phrase, "It's replaced 'Good morning' in my house."

But just when I'd about given up,
she turned around and said,
"Daddy, you teach all those
other guys to program, can you
teach me?"

"Like father, like daughter," says my mother. But she's wrong. I never had a cell phone.

It's as if when they hit 13, kids develop an inverter circuit that automatically negates whatever you tell them. I swear that if I said, "Annabelle, under no circumstances are you to stuff 47 tennis balls down the toilet," I'd get an eye roll, a roar I can't transliterate, and the automatic response, "That's not fair! All the other girls have 48. You're so mean. Please tell me I'm adopted." No such luck. Acorns and trees, kid.

But just when I'd about given up, she turned around and said, "Daddy, you teach all those other guys to program, can you teach me?"

I'd never really thought of that. Until last year, she attended a technology-averse Waldorf school, as do the kids of eBay's CTO (see nyti.ms/nupJKY). She learned to read and write with paper books (what a concept, see msdn.microsoft.com/magazine/jj863140), and to knit, which doesn't necessarily take you as far as reading and writing. I

see it as an inoculation against the myopic attitude that technology is everything—unlike the nearby Clark School, which trumpets the fact that their students learn PowerPoint in fourth grade. (God protect me from the PowerPoint presentations of fourth graders. Although they're probably better than some allegedly professional presenters I've endured, see msdn.microsoft.com/magazine/gg650665.)

Programming runs in my family. My mother programmed on a mainframe years ago, for her research in psychology at the time. I vaguely recall driving around Philadelphia, taking drawers of punch cards to the computer center to have them run, wondering what would happen if I switched just one or two. Nowadays, when my mother gets a condescending tech support person on the phone, she cuts him down to size by saying, "Look, I was programming mainframes before you were born. Now can the attitude and tell me how this thing works." I can envision Annabelle carrying on that tradition.

How should I teach her? What should I teach her? Managed or native? Objects or functions? Or maybe Visual Basic 6, so she can have a very long career (see msdn.microsoft.com/magazine/jj133828). Or perhaps a game—that's what first grabbed me years ago (text-based Star Trek, if you really want to know). I'd be curious to hear your ideas, dear readers.

I feel like Archie Manning, a stellar college quarterback who never had a winning season in the NFL. But he managed to pass his craft on to his sons Eli and Peyton, who have won three Super Bowls between them. I wouldn't mind seeing Annabelle surpass me.

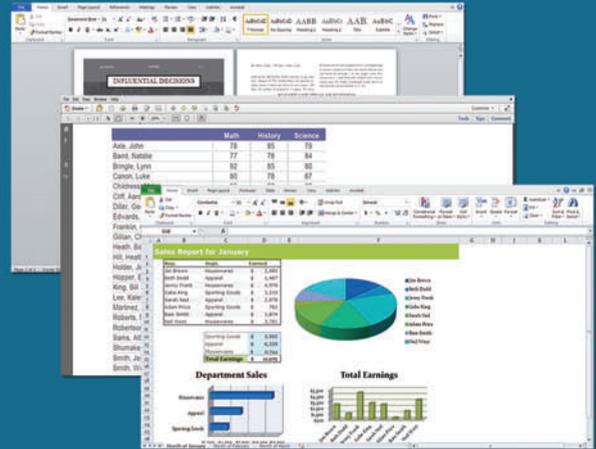
More than anything else, I'd tell her that the best developers are not the ones with the whizzingest-bangingest code, but rather the ones who can step outside the code and say, "Hey, what problem are we really trying to solve here?"

Every generation, at the height of its powers, needs to step aside for the next one. I hope that experience, wisdom and the occasional spot of treachery will keep me pouring oil on troubled fires for years yet. But I can see that day coming, one day at a time whether I'm looking or not, just as her 13th birthday did. I have the chance to shape it by shaping her. Now if I can just find the plumber's snake to unclog my toilet ... ■

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

WORKING WITH FILES?

CONVERT
PRINT
CREATE
COMBINE
& MODIFY



100% Standalone - No Office Automation



DOC, XLS, PPT, PDF, MSG, VSD, & image formats + many MORE

Get your FREE evaluation copy at www.aspose.com

US Sales: +1 888 277 6734
sales@aspose.com

EU Sales: +44 141 416 1112
sales.europe@aspose.com

 **ASPOSE**
Your File Format Experts

SCAN FOR
20% SAVINGS



New Essential Studio for JavaScript



The first JavaScript control framework designed for
line-of-business applications

Over 30 client-side controls designed from the ground up for
enterprise application development.

- ★ Exclusive **OLAP Grid**—Visualize business intelligence data on the web.
- ★ Powerful **Chart**—Visualize data in ways you didn't think possible on the client side.
- ★ Enhanced **Grid**—Employ a rich set of features, such as Microsoft Outlook-like grouping.
- ★ Stunning **Gauges**—Build client-side dashboards with ease.

Controls work on any platform—no IIS or specific back end required.

Download a free, 30-day evaluation today.

syncfusion.com/javascript

888-9-DOTNET



 Syncfusion®