THE MICROSOFT JOURNAL FOR DEVELOPERS

NOVEMBER 2011 VOL 26 NO 11

msdn magazine

COLUMNS

Microsoft

Write Once, Experience Many

check out infragistics.com/jquery

NetAdvantage® for jQuery

TREE

Simplify the look of hierarchical data, while offering the experience, design and functionality your users will love!

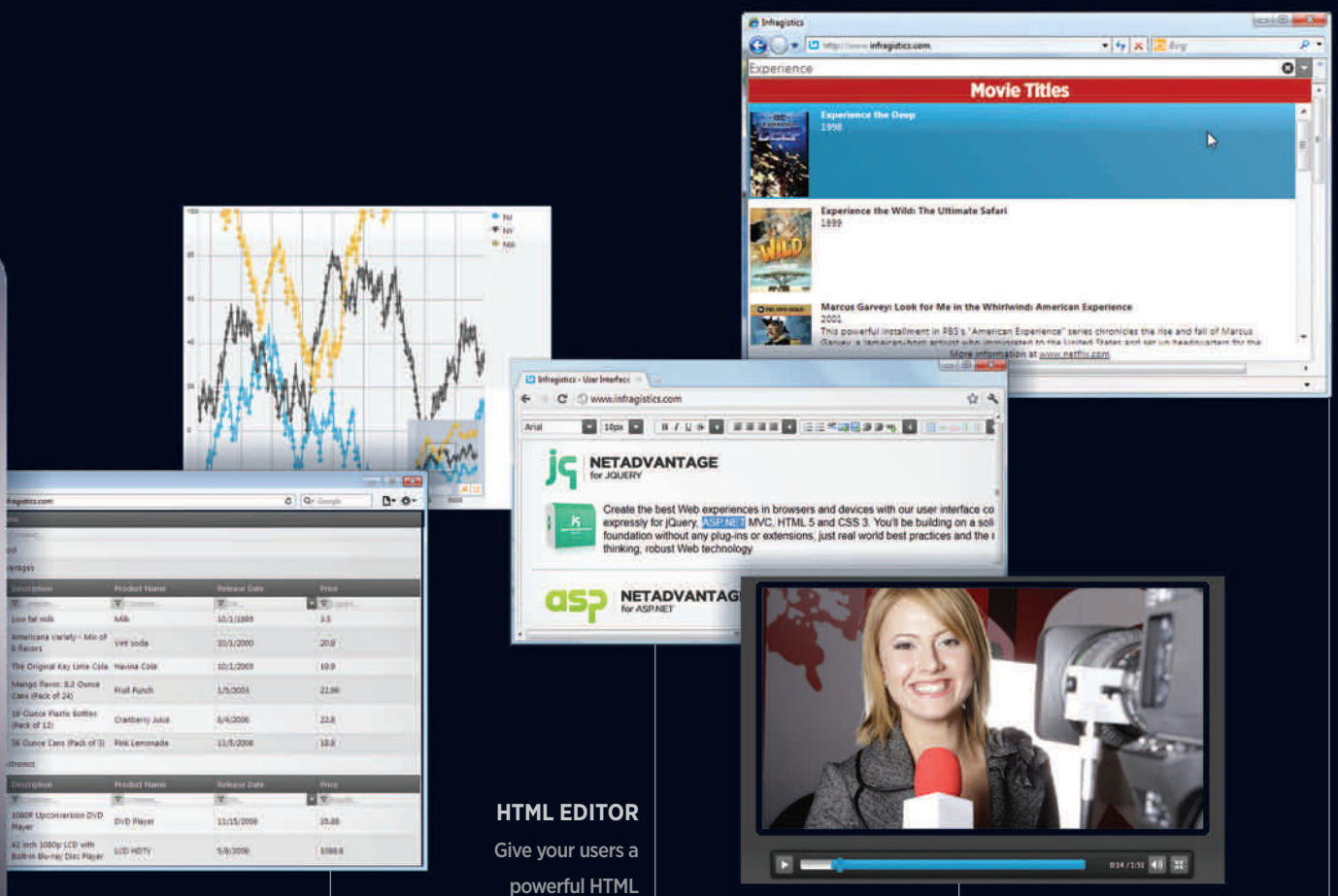
BUSINESS CHARTING

Combine interactive Outlook style grids with rich business charting to deliver a complete portable solution.

INFRAGISTICS

DESIGN / DEVELOP / EXPERIENCE

Infragistics Sales 800 231 8588 • Infragistics Europe Sales +44 (0) 800 298 9055 • Infragistics India +91 80 4151 8042 • @infragistics



HIERARCHICAL GRID

An expandable data grid that presents multiple parent-child relationships is the backbone of your data application.

HTML EDITOR

Give your users a powerful HTML editing experience by incorporating the jQuery WYSIWYG editing tool.

VIDEO PLAYER

When a user finds what they want to watch, our HTML5 video player adds streaming video right into your own apps.

COMBO

The fully featured combo box control offers intuitive auto-suggest, auto-complete and auto-filtering built in.

msdn magazine

NOVEMBER 2011 VOLUME 26 NUMBER 11

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.

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, Merit Direct. Phone: 914-368-1000; E-mail: 1105media@meritdirect.com; Web: www.meritdirect.com/1105

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

Microsoft

BPA WORLDWIDE

Printed in the USA

techxtend.com
866-719-1528

TechXtend | programmer's paradise

Embarcadero RAD Studio XE2

The ultimate application development suite for Windows, Mac, mobile and Web

by Embarcadero

Embarcadero® RAD Studio XE2 is the ultimate application development suite and the fastest way to build data-rich, visually engaging applications for Windows, Mac, mobile, .NET, PHP and the Web. RAD Studio includes Delphi®, C++Builder® and RadPHP™, enabling developers to deliver applications up to 5x faster across multiple desktop, mobile, Web, and database platforms including Delphi applications for 64-bit Windows.

NEW VERSION!

Professional Ed.
TechXtend # CGI 15501A01

$1,383.99

techxtend.com/embarcadero

UltraEdit

The world's #1 text editing solution is also the world's most affordable!

by IDM Computer Solutions

UltraEdit is the world's standard in text editors. Millions use UltraEdit as the ideal text/hex/programmers editor on any platform — Windows, Mac, or Linux!

Features include syntax highlighting for nearly any programming language; powerful Find, Replace, Find in Files, and Replace in Files; FTP support, sort, column mode, hex, macros/scripting, large file handling (4+ GB), projects, templates, Unicode, and more.

Named User
1-24 Users
TechXtend # I84 01201A01

$59.95

techxtend.com/idm

VMware vSphere 5 Essentials Kit

VMware vSphere is the industry-leading virtualization platform for building cloud infrastructures that enables users to run business critical applications with confidence and respond faster to their business.

NEW VERSION 5!

vSphere accelerates the shift to cloud computing for existing datacenters, while also underpinning compatible public cloud offerings paving the way for the only hybrid cloud model. With over 250,000 customers worldwide and the support of over 2500 applications from more than 1400 ISV partners, VMware vSphere is the trusted platform for any application.

for 3 hosts
TechXtend # V55 85201A01

$446.99

techxtend.com/vSphere

ActiveReports 6

by GrapeCity PowerTools

The de facto standard reporting tool for Microsoft Visual Studio .NET

- Fast and Flexible reporting engine
- Flexible event-driven API to completely control the rendering of reports
- Wide range of Export and Preview formats including Windows Forms Viewer, Web Viewer, Adobe Flash and PDF
- XCopy deployment
- Royalty-Free Licensing for Web and Windows applications

NEW VERSION 6!

Professional Ed.
TechXtend # D03 04301A01

$1,310.99

techxtend.com/grapecity

SlickEdit 2011 Bundle

by SlickEdit

All editors are not created equal. At SlickEdit, our belief is that it's the code that really matters. SlickEdit is designed for power programmers by power programmers. We take great pride in delivering unparalleled power, speed, and flexibility to our customers. Our goal is to remove the tedious tasks involved with programming, allowing you to focus on the reason you first got into programming: the thrill of writing great code.

- Speed — Use SlickEdit to write more code in less time. Spend less time waiting for results.
- Power — SlickEdit provides the power to work with your real-world projects.
- Flexibility — All programmers are different. SlickEdit lets you work the way you want.

Windows/Mac
TechXtend # M39 05501A01

$382.99

techxtend.com/slickedit

TX Text Control 16.0

Word Processing Components

TX Text Control is royalty-free, robust and powerful word processing software in reusable component form.

New Version Released!

- .NET WinForms and WPF rich text box for VB.NET and C#
- ActiveX for VB6, Delphi, VBScript/HTML, ASP
- File formats DOCX, DOC, RTF, HTML, XML, TXT
- PDF and PDF/A export, PDF text import
- Tables, headers & footers, text frames, bullets, structured numbered lists, multiple undo/redo, sections, merge fields, columns
- Ready-to-use toolbars and dialog boxes

Professional Edition
TechXtend # T79 12101A01

$1,109.99

Download a demo today.

techxtend.com/textcontrol

Microsoft Visual Studio Test Professional

by Microsoft

Make a sustainable impact to software quality with Microsoft Visual Studio Test Professional 2010, the integrated testing toolset that delivers a complete plan-test-track workflow. File high quality bugs with rich diagnostics for your developers. Take full advantage of a task-driven user interface and features like Fast Forward for Manual Testing so you can focus your time and energy on high-value tasks. With tight integration to Team Foundation Server you will gain in-context collaboration between all team roles, greatly increasing your visibility to the overall project.

with MSDN
TechXtend # M47 70101A01

$1,919.99

techxtend.com/microsoft

Lenovo ThinkPad X1

by Lenovo

Get introduced to the lightest corporate heavyweight in town. The ThinkPad X1 is the thinnest yet power-packed ThinkPad so far, built to deliver no-compromise performance. Powered by Intel Core processor, it lends superior VOIP experience with advanced noise and echo cancellation. Its Lenovo RapidCharge battery technology recharges batteries to 80% capacity in all of 30 minutes; while its trusted Roll Cage technology and Corning Gorilla glass make sure it stays protected against the rough and tumble of corporate life. Designed to match your sophistication and style, the ThinkPad X1 is your answer to uber-cool and high-performance mobile computing.

TechXtend # ZHI KD8043

$1,315.99

techxtend.com/lenovo

HP LaserJet P2035N

by Hewlett Packard

Enjoy fast print speeds, professional-quality output, and flexible connectivity options with the easy-to-use, value-packed HP LaserJet P2035 Printer series. Get your documents and get back to work quickly with print speeds of up to 30 pages per minute on letter size paper. Time to completion for a typical office print job can be nearly two times faster with the Instant-on Technology built into this printer. Quickly connect the printer to your computer with the Hi-Speed USB 2.

TechXtend # H22 03AW

$275.99

techxtend.com/hp

Flexera InstallShield 2012 Express

by Flexera Software

On more than 500 million PCs globally, InstallShield is the proven market leader and de facto industry standard installation software for authoring professional and reliable Windows Installer and InstallScript-based software installations and Microsoft App-V packages. Using InstallShield ensures that even the most complex software applications are correctly installed, configured, updated and uninstalled effectively on Windows desktops, servers, and mobile.

NEW VERSION 2012!

for Windows
TechXtend # I21H01501A01

$592.99

techxtend.com/flexera

Intel® Fortran Studio XE

by Intel

Intel® Fortran Studio XE suite combines Intel Fortran Composer XE — which includes scalable parallelism support through Fortran 2008, Co-Array Fortran, and Intel® Math Kernel Library — with error checking, code robustness, and performance profiling tools. The bundle includes three next generation industry-leading products: Intel® Fortran Composer XE, Intel® VTune™ Amplifier XE, and Intel® Inspector XE.

NEW!

Windows Upgrade
TechXtend # I23 92101S01

$1,205.99

techxtend.com/intel

top tech deals

$AVE, $AVE, $AVE!

TechXtend Top Tech Deals!

Check out our web site for special-priced offers!

Software, Systems, & Solutions – Since 1982

Prices subject to change. Not responsible for typographical errors.

EDITOR'S NOTE

MICHAEL DESMOND

A Game of Risk

When Microsoft announced the Windows Runtime (WinRT) stack at the heart of Windows 8 during the BUILD conference keynote in September, everyone in the room knew the game had changed.

In the decade since Microsoft launched the Microsoft .NET Framework and transitioned many developers to managed languages like C# and Visual Basic .NET, the company has adroitly leveraged its vast developer community. Every step along the way, Microsoft ushered its incumbent programmers forward with the promise of reusing existing code, working with familiar tools and exercising well-honed skills.

The strategy is both brilliant and obvious, and strangely mimics the late rounds of the board game Risk, when players inevitably spill enormous piles of armies onto the board. I've played my share of Risk and know well the incalculable glee that comes from cashing in a trifecta of cards for 60-plus armies. When you show up with numbers like that, things get *done* (yeah, I'm looking at you, Irkutsk).

The Trouble with Asia

The problem facing Microsoft, as any Risk player knows, is that even the massed armies of .NET developers couldn't, metaphorically speaking, hold Asia—that vast and vulnerable continent on the Risk board that has been the undoing of so many players. Smartphones, tablets and the emergence of HTML5 as a development target have created huge new frontiers—frontiers the Microsoft .NET strategy was simply not designed to address.

WinRT, however, is.

The WinRT stack at the heart of Windows 8 takes the .NET strategy and turns it inside out. Rather than urge developers to move into new languages, like Visual Basic .NET or C#, WinRT exposes its capabilities to multiple languages. By projecting the functionality of native Windows APIs into each language, Microsoft has thrown open the doors to a potentially huge community of developers across the C++, C#, Visual Basic .NET and JavaScript domains.

Microsoft is able to do this by implementing the APIs in a language-neutral fashion, including metadata that each language environment uses to "project" the APIs into its environment in a natural way. For example, at the lowest level, the WinRT APIs use HRESULTs for error reporting, but those errors are projected into languages like C# and JavaScript as exceptions.

The names of properties themselves also are specifically cased for each language, so a C++/C#/Visual Basic developer sees properties in Pascal-casing, while a JavaScript developer sees them in camel-casing, just like other APIs he already knows.

From C++ coders tuning multithreaded applications to JavaScript hobbyists hoping to create the next "Angry Birds," WinRT effectively broadens the definition of the phrase "Windows developer."

As one member of the WinRT development team told me: "The ability to write native apps in JavaScript/HTML opens up the world of Windows to a developer community that's probably an order of magnitude larger than the .NET developer community."

That noise you just heard was the sound of Microsoft slapping another set of Risk cards onto the board. That'll be 90 more armies, please.

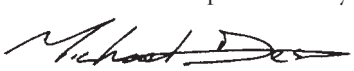
Taking the Board

Microsoft is articulating a path to extend WinRT even further.

"The beauty of the solution here is twofold," the WinRT team member said. "One is that the architecture easily allows for additional languages to be supported in the future, where that environment would again have immediate and direct access to native APIs. Two is that developers can also create their own APIs in this model—what we call WinRT components—such that they can plug into the language projections just as the native APIs do."

The result: "Hybrid" apps where the most appropriate language can be used for different parts of the software. So a math-intensive physics engine written in C++ can be utilized directly from, say, JavaScript.

Will Microsoft's bold strategy attract enough developers to help it win the board? As any Risk player knows, a lot depends on the roll of the dice. But it's clear that Microsoft is in a far better position today to address the challenge than it was just a few months ago.



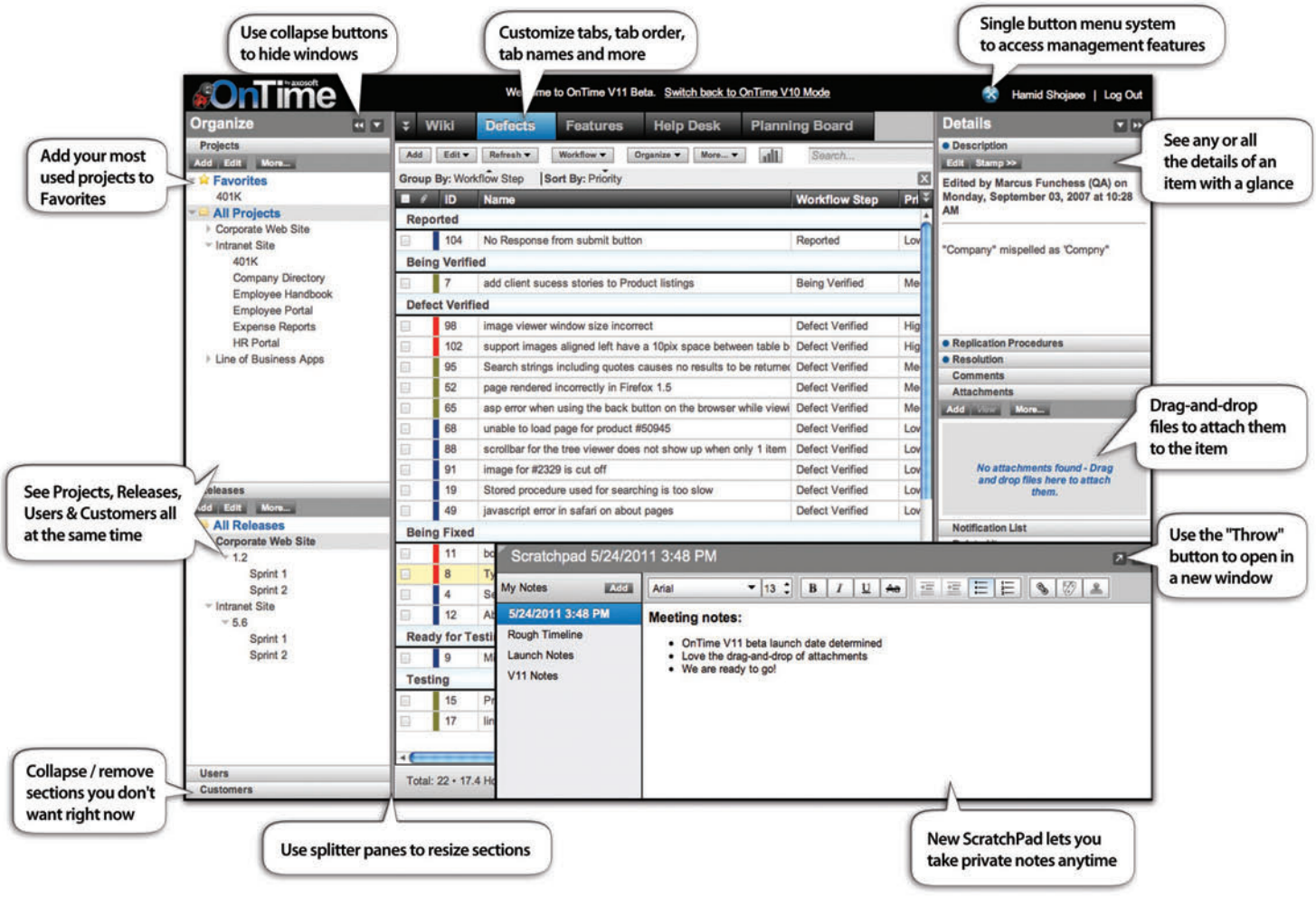
Visit us at msdn.microsoft.com/magazine. Questions, comments or suggestions for *MSDN Magazine*? Send them to the editor: mmeditor@microsoft.com.

OnTime11

axosoft.com
800.653.0024

Everything You Need to Ship Software OnTime

Agile Project Management • Bug Tracking • Backlogs • Burndowns • All in a Beautiful UI



Visit www.axosoft.com for the following:

FREE FOR 2 Users

Never expires, no credit card required. 2 users are totally free forever. Install on your servers or sign up for an account.

Making of OnTime 11

Watch the key principles, the design philosophy and what separates OnTime from other competing products.

Join a Live Demo

Refreshingly informative, and certainly not a sales pitch. We offer 3 live demos a week, so reserve your spot today!

DINO ESPOSITO

Design of a Domain Model

The recent release of the Entity Framework 4.1 and the new Code First development pattern breaks a cardinal rule of server development: Don't take a single step if the database isn't in place. Code First tells developers to focus on the business domain and to model it in terms of classes. In a way, Code First encourages the application of domain-driven design (DDD) principles in the .NET space. A business domain is populated with related, interconnected entities, each of which has its own data exposed as properties and may expose behavior through methods and events. More importantly, each entity may have a state and be bound to a possibly dynamic list of validation rules.

Writing an object model for a realistic scenario raises some issues that aren't addressed in current demos and tutorials. In this article, I'll take the challenge and discuss the building of a Customer class, touching on a number of design patterns and practices along the way, such as the Party pattern, aggregate roots, factories, and technologies like Code Contracts and the Enterprise Library Validation Application Block (VAB).

For reference, I recommend you take a look at an open source project of which the code discussed here is a small subset. Created by Andrea Saltarello, the Northwind Starter Kit project (nsk.codeplex.com) aims to illustrate effective practices for architecting multilayered solutions.

Object Model vs. Domain Model

Debating whether to employ an object model or a domain model may seem pointless, and for the most part it's just a matter of terminology. But precise terminology is a key factor in ensuring that all members of a team have the same concept in mind when they use specific terms.

For nearly everybody in the software industry, an object model is a collection of generic but possibly related objects. How is a domain model different? In the end, a domain model is still an object model, so using the two terms interchangeably may not be a terrible mistake. Still, when the term "domain model" is used with a certain emphasis, it may carry with it some expectations about the shape of constituent objects.

This use of domain model is associated with the definition given by Martin Fowler: an object model of the domain that incorporates both behavior and data. In turn, the behavior expresses both rules and specific logic (see bit.ly/60l6uQ).

DDD adds a bunch of pragmatic rules to a domain model. According to this perspective, a domain model differs from an object model in the intensive use of value objects it recommends in lieu of primitive types. An integer, for example, can be many things—a temperature, an amount of money, a size, a quantity. A domain model would use a specific value object type for each different scenario.

Furthermore, a domain model should identify aggregate roots. An aggregate root is an entity obtained by composing other entities together. Objects in the aggregate root have no relevance outside,

Figure 1 **Classes According to the Party Pattern**

```
public abstract class Party
{
  public virtual String Name { get; set; }
  public virtual PostalAddress MainPostalAddress { get; set; }
}
public abstract class Person : Party
{
  public virtual String Surname { get; set; }
  public virtual DateTime BirthDate { get; set; }
  public virtual String Ssn { get; set; }
}
public abstract class Organization : Party
{
  public virtual String VatId { get; set; }
}
```

Figure 2 **Customer Class as an Aggregate Root**

```
public class Customer : Organization, IAggregateRoot
{
  public static Customer CreateNewCustomer(
    String id, String companyName, String contactName)
  {
    ...
  }

  protected Customer()
  {
  }

  public virtual String Id { get; set; }
    ...

  public virtual IEnumerable<Order> Orders
  {
    get { return _Orders; }
  }

  Boolean IAggregateRoot.CanBeSaved
  {
    get { return IsValidForRegistration; }
  }

  Boolean IAggregateRoot.CanBeDeleted
  {
    get { return true; }
  }
}
```

Blazing-Fast **GRID CONTROLS**

Optimized for .NET

Award-Winning Presentation Controls and Reporting Libraries

Learn more and download your **FREE** evaluation copy today

Visit **DEVEXPRESS.COM/GRIDS**

***or Call Us* (818) 844-3383**

DevExpress™

PRESENTATION CONTROLS | REPORTING CONTROLS
BUSINESS APP FRAMEWORKS | IDE PRODUCTIVITY TOOLS

meaning that no use cases exist in which they're used without being passed from the root object. The canonical example of an aggregate root is the Order entity. Order contains as an aggregate OrderItem, but not Product. It's hard to imagine (even though that would be determined only by your specs) that you'd need to work with an OrderItem without it coming from an Order. On the other hand, you may well have use cases in which you work with Product entities without involving orders. Aggregate roots are responsible for maintaining their child objects in a valid state and persisting them.

Finally, some domain model classes may offer public factory methods to create new instances, rather than constructors. When the class is mostly standalone and not really part of a hierarchy, or when the steps that create the class are of some interest to the client, using a plain constructor is acceptable. With complex objects like aggregate roots, however, you need an additional level of abstraction over the instantiation. DDD introduces factory objects (or, more simply, factory methods on some classes) as a way to decouple client requirements from internal objects and their relationships and rules. A very clear and concise introduction to DDD can be found at bit.ly/oxoJD9.

The Party Pattern

Let's focus on a Customer class. In light of what was stated earlier, here's a possible signature:

```
public class Customer : Organization, IAggregateRoot
{
  ...
}
```

Who is your customer? Is it an individual, an organization or both? The Party pattern suggests you distinguish between the two and define clearly which properties are common and which belong only to individuals or organizations. The code in **Figure 1** is limited to Person and Organization; you can make it more detailed by splitting organizations into non-profit and commercial companies if your business domain requires it.

It's never a bad idea to recall that you should aim to produce a model that closely models your actual business domain, not an abstract representation of the business. If your requirements only speak of customers as individuals, then applying the Party pattern is not strictly required even though it introduces a point of future extensibility.

Customer as an Aggregate Root Class

An aggregate root is a class in your model that represents a standalone entity—one that doesn't exist in relation to other entities. Most of the time, you have aggregate roots that are just individual classes that don't manage any child object or perhaps simply point to the root of other aggregates. **Figure 2** shows a bit more of the Customer class.

As you can see, the Customer class implements the (custom) IAggregateRoot interface. Here's the interface:

```
public interface IAggregateRoot
{
  Boolean CanBeSaved { get; }
  Boolean CanBeDeleted { get; }
}
```

What does it mean to be an aggregate root? An aggregate root handles persistence for its child aggregated objects and is responsible for enforcing invariant conditions that involve the group. It turns out that an aggregate root should be able to check whether the entire stack can be saved or deleted. A standalone aggregate root just returns true without any further checking.

Figure 3 **Factory Method on the Customer Class**

```
public static Customer CreateNewCustomer(
  String id, String companyName, String contactName)
{
  Contract.Requires<ArgumentNullException>(
          id != null, "id");
  Contract.Requires<ArgumentException>(
          !String.IsNullOrWhiteSpace(id), "id");
  Contract.Requires<ArgumentNullException>(
          companyName != null, "companyName");
  Contract.Requires<ArgumentException>(
          !String.IsNullOrWhiteSpace(companyName), "companyName");
  Contract.Requires<ArgumentNullException>(
          contactName != null, "contactName");
  Contract.Requires<ArgumentException>(
          !String.IsNullOrWhiteSpace(contactName), "contactName");

  var c = new Customer
              {
                Id = id,
                Name = companyName,
                  Orders = new List<Order>(),
                ContactInfo = new ContactInfo
                              {
                                 ContactName = contactName
                              }
              };
  return c;
}
```

Factory and Constructor

A constructor is type-specific. If the object is just one type—no aggregates and no complex initialization logic—using a plain constructor is more than fine. In general, however, a factory is a useful extra layer of abstraction. A factory can be a simple static method on the entity class or a separate component of its own. Having a factory method also helps readability, because it makes clear why you're creating that given instance. With constructors, your power to address different instantiation scenarios is more limited, as constructors aren't named methods and can only be distinguished through the signature. Especially with long signatures, it's difficult to figure out later why a particular instance is being obtained. **Figure 3** shows the factory method on the Customer class.

A factory method is atomic, gets input parameters, does its job and returns a fresh instance of a given type. The instance being returned should be guaranteed to be in a valid state. The factory is responsible for fulfilling all defined internal validation rules.

The factory also needs to validate input data. For this, using Code Contracts preconditions keeps the code clean and highly readable. You can also use postconditions to ensure that the returned instance is in a valid state, like so:

```
Contract.Ensures(Contract.Result<Customer>().IsValid());
```

As for using invariants throughout the class, experience says that you can't always afford them. Invariants may be too invasive, especially in large, complex models. Code Contracts invariants are sometimes almost too respectful of the ruleset, and sometimes in your code you want more flexibility. It's preferable, then, to restrict the areas where invariants must be enforced.

Rock-Solid **REPORTING CONTROLS**
Optimized for.NET

Award-Winning Presentation Controls and Reporting Libraries

Learn more and download your **FREE** evaluation copy today
Visit **DEVEXPRESS.COM/REPORTING**
***or Call Us* (818) 844-3383**

DevExpress™

PRESENTATION CONTROLS | REPORTING CONTROLS
BUSINESS APP FRAMEWORKS | IDE PRODUCTIVITY TOOLS

Figure 4 **Enterprise Library Rulesets**

```
<validation>
  <type assemblyName="..." name="ValidModel1.Domain.Customer">
    <ruleset name="IsValidForRegistration">
      <properties>
        <property name="CompanyName">
          <validator negated="false"
                     messageTemplate="The company name cannot be null"
                     type="NotNullValidator" />
          <validator lowerBound="6" lowerBoundType="Ignore"
                     upperBound="40" upperBoundType="Inclusive"
                     negated="false"
                     messageTemplate="Company name cannot be longer ..."
                     type="StringLengthValidator" />
        </property>
        <property name="Id">
          <validator negated="false"
                     messageTemplate="The customer ID cannot be null"
                     type="NotNullValidator" />
        </property>
        <property name="PhoneNumber">
          <validator negated="false"
                     type="NotNullValidator" />
          <validator lowerBound="0" lowerBoundType="Ignore"
                     upperBound="24" upperBoundType="Inclusive"
                     negated="false"
                     type="StringLengthValidator" />
        </property>
        <property name="FaxNumber">
          <validator negated="false"
                     type="NotNullValidator" />
          <validator lowerBound="0" lowerBoundType="Ignore"
                     upperBound="24" upperBoundType="Inclusive"
                     negated="false"
                     type="StringLengthValidator" />
        </property>
      </properties>
    </ruleset>
  </type>
</validation>
```

Validation

Properties on a domain class likely need to be validated to ensure that no required fields are left empty, no too-long text is placed in limited containers, values fall in the proper ranges and so forth. You'll also have to consider cross-property validation and sophisticated business rules. How would you code validation?

Validation is about conditional code so, in the end, it's a matter of combining a few if statements and return Booleans. Writing a validation layer with plain code and without any framework or technology might work, but it isn't really a great idea. The resulting code wouldn't be very readable and wouldn't be easy to evolve, though some fluent libraries are making this easier. Subject to real business rules, validation can be highly volatile and your implementation must account for that. In the end, you can't simply write code that validates; you have to write code that's open to validating the same data against different rules.

With validation, sometimes you want to yell out if invalid data is passed, and sometimes you just want to collect errors and report that to other layers of code. Remember, Code Contracts don't validate; they check conditions and then throw an exception if a condition doesn't apply. By using a centralized error handler you can recover from exceptions and degrade gracefully. In general, I recommend using Code Contracts in a domain entity only to catch potentially severe errors that can lead to inconsistent states. It makes sense to use Code Contracts in a factory—in this case, if passed data is invalid, the code must throw. Whether to use Code Contracts in the setter methods of properties is your call. I prefer to take a softer route and validate via attributes. But which attributes?

Data Annotations vs. VAB

The Data Annotations namespace and Enterprise Library VAB are very similar. Both frameworks are attribute-based and can be extended with custom classes representing custom rules. In both cases, you can define cross-property validation. Finally, both frameworks have a validator API that evaluates an instance and returns the list of errors. Where's the difference?

Data Annotations are part of the Microsoft .NET Framework and don't require a separate download. Enterprise Library is a separate download; not a big deal in a large project, but still an issue as it may require approval in corporate scenarios. Enterprise Library can be easily installed via NuGet (see the article, "Manage Project Libraries with NuGet," in this issue).

The Enterprise Library VAB is superior to Data Annotations in one respect: It can be configured via XML rulesets. An XML ruleset is an entry in the configuration file where you describe the validation you want. Needless to say, you can change things declaratively without even touching your code. **Figure 4** shows a sample ruleset.

A ruleset lists the attributes you want to apply to a given property on a given type. In code, you validate a ruleset as follows:

```
public virtual ValidationResults ValidateForRegistration()
{
  var validator = ValidationFactory
          .CreateValidator<Customer>("IsValidForRegistration");
  var results = validator.Validate(this);
  return results;
}
```

The method applies the validators listed in the IsValidForRegistration ruleset to the specified instance.

A final note on validation and libraries. I haven't covered every popular validation library, but that wouldn't make a significant difference. The important point is to consider whether your business rules change and how often. Based on that, you can decide whether Data Annotations, VAB, Code Contracts or some other library is more appropriate. In my experience, if you know exactly what you need to achieve, then the "right" validation library is easy to choose.

Wrapping Up

An object model for a realistic business domain can hardly be a plain collection of properties and classes. Moreover, design considerations take precedence over technologies. A well-done object model expresses every necessary aspect of the domain. Most of the time, this means having classes that are easy to initialize and validate, and are rich in properties and logic. DDD practices should not be considered dogmatically, but instead be the guideposts that show the way to go. ■

DINO ESPOSITO *is the author of "Programming Microsoft ASP.NET 4" (Microsoft Press, 2011) and "Programming Microsoft ASP.NET MVC" (Microsoft Press, 2011), and coauthor of "Microsoft .NET: Architecting Applications for the Enterprise" (Microsoft Press, 2008). Based in Italy, Esposito is a frequent speaker at industry events worldwide. Follow him on Twitter at twitter.com/despos.*

THANKS to the following technical experts for reviewing this article: *Manuel Fahndrich and Andrea Saltarello*

Powerhouse **ANALYTICS**

Optimized for.NET

Award-Winning Presentation Controls and Reporting Libraries

Learn more and download your **FREE** evaluation copy today

Visit **DEVEXPRESS.COM/ANALYTICS**

***or Call Us* (818) 844-3383**

DevExpress™

PRESENTATION CONTROLS | REPORTING CONTROLS
BUSINESS APP FRAMEWORKS | IDE PRODUCTIVITY TOOLS

KENNY KERR

Thread Pool Synchronization

I've said it before: Blocking operations are bad news for concurrency. Often, however, you need to wait for some resource to become available, or perhaps you're implementing a protocol that stipulates some amount of time needs to elapse before resending a network packet. What do you do then? You could use a critical section, call functions like Sleep and WaitForSingleObject and so on. Of course, that means you'll have threads just sitting around blocking again. What you need is a way for the thread pool to wait on your behalf without affecting its concurrency limits, which I discussed in my September 2011 column (msdn.microsoft.com/magazine/hh394144). The thread pool can then queue a callback once the resource is available or the time has elapsed.

In this month's column, I'm going to show you how you can do just that. Along with work objects, which I introduced in my August 2011 column (msdn.microsoft.com/magazine/hh335066), the thread pool API provides a number of other callback-generating objects. This month, I'm going to show you how to use wait objects.

Wait Objects

The thread pool's wait object is used for synchronization. Rather than block on a critical section—or slim reader/writer lock—you can wait for a kernel synchronization object, commonly an event or semaphore, to become signaled.

Although you can use WaitForSingleObject and friends, a wait object integrates nicely with the rest of the thread pool API. It does this quite efficiently by grouping together any wait objects that you submit, reducing the number of required threads and the amount of code you need to write and debug. This allows you to use a thread pool environment and cleanup groups, and also frees you from having to dedicate one or more threads to wait for objects to become signaled. Due to improvements in the kernel portion of the thread pool, it can in some cases even achieve this in a threadless manner.

The CreateThreadpoolWait function creates a wait object. If the function succeeds, it returns an opaque pointer representing the wait object. If it fails, it returns a null pointer value and provides more information via the GetLastError function. Given a work object, the CloseThreadpoolWait function informs the thread pool that the object may be released. This function doesn't return a value, and for efficiency assumes the wait object is valid.

The unique_handle class template I introduced in my July 2011 column (msdn.microsoft.com/magazine/hh288076) takes care of these details.

Here's a traits class that can be used with unique_handle, as well as a typedef for convenience:

```
struct wait_traits
{
  static PTP_WAIT invalid() throw()
  {
    return nullptr;
  }

  static void close(PTP_WAIT value) throw()
  {
    CloseThreadpoolWait(value);
  }
};

typedef unique_handle<PTP_WAIT, wait_traits> wait;
```

I can now use the typedef and create a wait object as follows:

```
void * context = ...
wait w(CreateThreadpoolWait(callback, context, nullptr));
check_bool(w);
```

As usual, the final parameter optionally accepts a pointer to an environment so that you can associate the wait object with an environment, as I described in my September column. The first parameter is the callback function that will be queued to the thread pool once the wait completes. The wait callback is declared as follows:

```
void CALLBACK callback(PTP_CALLBACK_INSTANCE, void * context, PTP_WAIT,
TP_WAIT_RESULT);
```

The callback TP_WAIT_RESULT argument is just an unsigned integer providing the reason why the wait completed. A value of WAIT_OBJECT_0 indicates that the wait was satisfied as the synchronization object became signaled. Alternatively, a value of WAIT_TIMEOUT indicates that the timeout interval elapsed before the synchronization object was signaled. How would you indicate the timeout and synchronization object to wait for? That's the job of the surprisingly complex SetThreadpoolWait function. This function is simple enough until you try to specify a timeout. Consider this example:

```
handle e(CreateEvent( ... ));
check_bool(e);

SetThreadpoolWait(w.get(), e.get(), nullptr);
```

First, I create an event object, using the unique_handle typedef from my July column. Not surprisingly, the SetThreadpoolWait function sets the synchronization object that the wait object is to wait for. The last parameter indicates an optional timeout, but in this example, I provide a null pointer value, indicating that the thread pool should wait indefinitely.

The FILETIME Structure

But what about a specific timeout? That's where it gets tricky. Functions such as WaitForSingleObject let you set a timeout value

Elegant **CHARTING CONTROLS**

Optimized for .NET

Award-Winning Presentation Controls and Reporting Libraries

Learn more and download your **FREE** evaluation copy today

Visit **DEVEXPRESS.COM/CHARTING**

***or Call Us* (818) 844-3383**

DevExpress™

PRESENTATION CONTROLS | REPORTING CONTROLS
BUSINESS APP FRAMEWORKS | IDE PRODUCTIVITY TOOLS

in milliseconds as an unsigned integer. The SetThreadpoolWait function, however, expects a pointer to a FILETIME structure, which presents a few challenges to the developer. The FILETIME structure is a 64-bit value representing an absolute date and time since the beginning of the year 1601 in 100-nanosecond intervals (based on Coordinated Universal Time).

To accommodate relative time intervals, SetThreadpoolWait treats the FILETIME structure as a signed 64-bit value. If a negative value is provided, it takes the unsigned value as a time interval relative to the current time, again in 100-nanosecond intervals. It's worth mentioning that the relative timer stops counting when the computer is sleeping or hibernating. Absolute timeout values are obviously not affected by this. Anyway, this use of FILETIME is not convenient for either absolute or relative timeout values.

Probably the simplest approach for absolute timeouts is to fill out a SYSTEMTIME structure and then use the SystemTimeToFileTime function to prepare a FILETIME structure for you:

```
SYSTEMTIME st = {};
st.wYear = ...
st.wMonth = ...
st.wDay = ...
st.wHour = ...
// etc.

FILETIME ft;
check_bool(SystemTimeToFileTime(&st, &ft));

SetThreadpoolWait(w.get(), e.get(), &ft);
```

For relative timeout values, a bit more thinking is involved. First, you need to convert some relative time into 100-nanosecond intervals, and then convert it to a negative 64-bit value. The latter is trickier than it seems. Remember that computers represent signed integers using the two's complement system, with the effect that a negative value must have its most significant bit set high. Added to this is the fact that FILETIME actually consists of two 32-bit values. This means you also need to handle machine alignment properly when treating it as a 64-bit value, otherwise an alignment fault may occur. Additionally, you can't simply use the lower 32-bit values to store the value, as the most significant bit is in the higher 32-bit values.

Relative Timeout Value Conversion

It's common to express relative timeouts in milliseconds, so let me demonstrate that conversion here. Recall that a millisecond is a thousandth of a second and a nanosecond is a billionth of a second. Another way to look at it is that a millisecond is 1,000 microseconds and a microsecond is 1,000 nanoseconds. A millisecond is then 10,000 100-nanosecond units, the unit of measurement expected by SetThreadpoolWait. There are many ways to express this, but here's one approach that works:

```
DWORD milliseconds = ...
auto ft64 = -static_cast<INT64>(milliseconds) * 10000;

FILETIME ft;
memcpy(&ft, &ft64, sizeof(INT64));

SetThreadpoolWait(w.get(), e.get(), &ft);
```

Notice that I'm careful to cast the DWORD before the multiplication to avoid integer overflow. I also use memcpy, because a reinterpret_cast would require the FILETIME to be aligned on an 8-byte boundary. You could, of course, do that instead, but this is a bit cleaner. An even simpler approach takes advantage of the fact that the Visual C++ compiler aligns a union with the largest alignment requirement of any of its members. In fact, if you order the union members correctly, you can do it in just one line, as follows:

```
union FILETIME64
{
  INT64 quad;
  FILETIME ft;
};

FILETIME64 ft = { -static_cast<INT64>(milliseconds) * 10000 };

SetThreadpoolWait(w.get(), e.get(), &ft.ft);
```

Enough compiler tricks. Let's get back to the thread pool. Another thing you might be tempted to try is a zero timeout. This is commonly done using WaitForSingleObject as a way to determine whether a synchronization object is signaled without actually blocking and waiting. However, this procedure isn't supported by the thread pool, so you're better off sticking with WaitForSingleObject.

If you want a particular work object to cease waiting for its synchronization object, then simply call SetThreadpoolWait with a null pointer value as its second parameter. Just watch out for the obvious race condition.

> Another thing you might be tempted to try is a zero timeout.

The final function related to wait objects is WaitForThreadpoolWaitCallbacks. At first, it may appear similar to the WaitForThreadpoolWorkCallbacks function used with work objects, which I introduced in my August column. Don't let it fool you. The WaitForThreadpoolWaitCallbacks function literally does what its name suggests. It waits for any callbacks from the particular wait object.

The catch is that the wait object will only queue a callback when either the associated synchronization object is signaled or the timeout expires. Until one of those events occurs, no callbacks are queued and there's nothing for the Wait function to wait for. The solution is to first call SetThreadpoolWait with null pointer values, telling the wait object to cease waiting, and then call WaitForThreadpoolWaitCallbacks to avoid any race conditions:

```
SetThreadpoolWait(w.get(), nullptr, nullptr);
WaitForThreadpoolWaitCallbacks(w.get(), TRUE);
```

As you might expect, the second parameter determines whether any pending callbacks that may have slipped through but have not yet begun to execute will be canceled or not. Naturally, wait objects work well with cleanup groups. You can read my October 2011 (msdn.microsoft.com/magazine/hh456398) column to find out how to use cleanup groups. On larger projects, they really do help simplify a lot of the trickier cancelation and cleanup that needs to be done. ■

KENNY KERR *is a software craftsman with a passion for native Windows development. Reach him at kennykerr.ca.*

THANKS to the following technical expert for reviewing this article: *Pedro Teixeira*

Kendo UI

THE ART OF WEB DEVELOPMENT

Everything You Need

For JavaScript & HTML5 development

Kendo UI is a complete framework for JavaScript developers. It provides everything you need to build HTML5 and JavaScript sites and apps, including a powerful data source, crazy-fast templating, rich UI widgets, and customizable themes. Don't waste time piecing together a JavaScript framework. Download Kendo UI and start developing amazing JavaScript apps right away.

Download the future of JavaScript development
at **www.kendoui.com** or scan

JULIE LERMAN

What the Heck Are Document Databases?

There's a good chance that you've at least heard of the term NoSQL by now. Articles have even been written about it here in *MSDN Magazine*. A lot of people who I highly respect are quite excited about it, and having grown up on relational databases, I wanted to have a better understanding of the space. I've done quite a bit of research and pestering of friends to wrap my head around it, and here I'll share what I've learned about a subset of NoSQL databases called "document databases." Another subset is key-value pair databases. Windows Azure Table Storage, which I wrote about in my July 2010 Data Points column (msdn.microsoft.com/magazine/ff796231), is an example of a key-value pair NoSQL store.

I should first address the definition of NoSQL. It's become a bit of a ubiquitous and possibly overused term. The term is used to encompass data storage mechanisms that aren't relational and therefore don't require using SQL for accessing their data. In his blog post, "Addressing the NoSQL Criticism" (bit.ly/rkphh0), CouchDB expert and author Bradley Holt says that he's heard people "redefining NoSQL as 'not only SQL.'" His point is that this isn't an *anti-SQL* movement by any means. I like this perspective, because I'm a big believer in using the right tool for the job.

Most databases that fall under the nonrelational umbrella share common goals of speed and scalability. By breaking away from the relational storage model and leaving schemas behind, these databases are free of the limitations put upon them by a tightly bound schema and your application's need to join data across tables.

Of the many document databases available, I'll focus on two of the most popular—MongoDB (mongodb.org) and CouchDB (couchdb.apache.org)—as well as RavenDB (ravendb.net), which was written for the Microsoft .NET Framework and is growing in popularity (see the article, "Embedding RavenDB into an ASP.NET MVC 3 Application," in this issue). This will remain high-level, though you can learn many more details about the individual databases and what makes them unique from one another by visiting their Web sites.

With the exception of a few twists (which I'll point out in this article), these databases provide their data most commonly through HTTP, store their data as JavaScript Object Notation (JSON) documents and provide APIs in multiple languages. The overall concerns are simplicity, speed and scalability. Equally important is that all three are open source projects.

In my research, I've heard a MongoDB expert say that the product's primary concern is performance. A CouchDB expert pointed to simplicity and reliability ("we want to be the Honda Accord of databases"). And Ayende Rahien, creator of RavenDB, said RavenDB aims for "fast writes, fast reads and world peace." Each of these document databases has even more to offer than what these sound bites suggest.

An Alternative, Not a Replacement, for Relational Databases

The NoSQL and document databases provide an alternative to relational databases, not a replacement. Each has its place, and they simply provide you with more options from which to choose. But how to choose? An important gauge is the Consistency, Availability and Partition Tolerance (CAP) theorem. It says that when working in distributed systems, you can only have two of the three guarantees (the C, the A or the P), so you have to pick what's important. If Consistency is the most critical, then you need to go with a relational database.

A common example of where Consistency would be the most important guarantee is in a banking application or perhaps one that runs a nuclear facility. In these scenarios, it's critical that every single piece of data is accounted for at every moment. If someone makes a withdrawal, you really need to know about it when you're looking at his account balance. Therefore, you'll probably want a relational database with a high level of control over its transactions. A term you'll hear frequently is "eventual consistency," or as expressed on the RavenDB site: "better stale than offline." In other domains, eventual consistency is sufficient. It's OK if data you're retrieving isn't up-to-the-millisecond accurate.

Perhaps, then, it's more important that some version of the data is available, rather than waiting for all of the transactions to catch up. This is related to the A (Availability) in CAP, which is focused on server uptime. Knowing that you'll always have access to the database takes precedence and is a huge benefit to database performance (that is, document databases are fast!). You'll find that the P, Partition Tolerance, is also important to the document databases, especially when scaling horizontally.

RESTful HTTP API—Mostly

Many of the NoSQL databases are accessible in a RESTful way, so you make your database connection through a URI, and the queries and commands are HTTP calls. MongoDB is an exception. Its default is to use TCP for database interactions, although there's at least one HTTP API available, as well. CouchDB and MongoDB provide language-specific APIs that let you write and execute queries and updates without having to worry about writing the HTTP calls directly. RavenDB has a .NET client API that simplifies interacting with the database.

Related Data in a Single Record

A lot of people incorrectly presume that nonrelational databases are flat files. The documents stored in a document database are capable of containing shaped data: trees with nodes. Each record in the database is a document and can be an autonomous set of data. It's self-describing—including its possibly unique schema—and isn't necessarily dependent on any other document.

Following is a typical example of what a record might look like in a document database (I'll steal a sample from the MongoDB tutorial that represents a student):

```
{
  "name" : "Jim",
  "scores" : [ 75, 99, 87.2 ]
}
```

And here's one from the CouchDB introductory article, which describes a book:

```
{
  "Subject": "I like Plankton"
  "Author": "Rusty"
  "PostedDate": "5/23/2006"
  "Tags": ["plankton", "baseball", "decisions"]
  "Body": "I decided today that I don't like baseball. I like plankton."
}
```

These are simple structures with string data, numbers and arrays. You can also embed objects within objects for a more complex document structure, such as this blog post example:

```
{
  "BlogPostTitle": "LINQ Queries and RavenDB",
  "Date":"\/Date(1266953391687+0200)\/",
  "Content":"Querying RavenDB is very familiar for .NET developers who are already
    using LINQ for other purposes",
  "Comments":[
             {
             "CommentorName":"Julie",
             "Date":"\/Date(1266952919510+0200)\/",
             "Text":"Thanks for using something I already know how to
               work with!",
             "UserId":"users/203907"
             },
  ]
}
```

Unique Keys

All of the databases require a key. If you don't provide one, they'll create one internally for you. Keys are critical to the databases' ability to index, but your own domain may require that you have known keys. In the previous blog post example, notice that there's a reference to "users/203907." This is how RavenDB leverages key values and allows you to define relationships between documents.

Storage in JSON Format

What these sample records all have in common is that they're using JSON to store their data. CouchDB and RavenDB (and many others) do in fact store their data in JSON. MongoDB uses a twist on JSON called Binary JSON (BSON) that's able to perform binary serialization. BSON is the internal representation of the data, so from a programming perspective, you shouldn't notice any difference.

The simplicity of JSON makes it easy to transpose object structures of almost any language into JSON. Therefore, you can define your objects in your application and store them directly in the database. This relieves developers of the need to use an object-relational mapper (ORM) to constantly translate between the database schema and the class/object schema.

Full-text searching engines—such as Lucene (lucene.apache.org), which is what RavenDB relies on—provide high-performance searching on this text-based data.

Notice the date in the blog post example. JSON doesn't have a date type, but each of the databases provides a way to interpret date types from whichever language you're coding in. If you check out the Data Types and Conventions list for the MongoDB BSON API (bit.ly/o87Gnx), you'll see that a date type is added, along with a few others, to flesh out what's available in JSON.

Storing and retrieving related data in a single unit can have huge performance and scalability benefits. Databases don't have to go trolling around to find data that's commonly related, because it's all together.

Collections of Types

When interacting with the database, how does your application know that one item is a student, another is a book and another is a blog post? The databases use a concept of collections. Any document, regardless of its schema, that's associated with a particular collection—for example, a student collection—can be retrieved when requesting data from that collection. It's also not uncommon to use a field to indicate type. This just makes searches a lot easier, but it's up to your application to enforce what should and shouldn't go into a collection.

Schema-Less Database

The "student" described earlier contains its own schema. Each record is responsible for its own schema, even those contained in a single database or collection. And one student record doesn't necessarily need to match another student record. Of course, your software will need to accommodate any differences. You could simply leverage this flexibility for efficiency. For example, why store null values? You could do the following when a property, such as "most_repeated class," has no value:

```
"name" : "Jim",
"scores" : [ 75, 99, 87.2 ]
"name" : "Julie",
"scores" : [ 50, 40, 65 ],
"most_repeated_class" : "Time Management 101"
```

Yes, Virginia, We Do Support Transactions

Each of the databases provides some level of transaction support—some more than others—but none are as rich as what can be achieved in a relational database. I'll defer to their documentation and let you follow up with your own additional research.

Document Databases and Domain-Driven Development

One of the core concepts of domain-driven development relates to modeling your domain using aggregate roots. When planning your domain classes (which may become the documents in your database), you can look for data that's most often self-contained (for example, an order with its line items) and focus on that as an individual data structure. In an ordering system, you'll probably also have customers and products. But an order might be accessed without needing its customer information and a product might be used without needing

access to the orders in which it's used. This means that although you'll find many opportunities to have self-contained data structures (such as the order with its line items), that doesn't rule out the need or capability to join data through foreign keys in certain scenarios.

Each of the databases provides guidance on various patterns that are available and which ones their users are having the most success with. For example, MongoDB documentation talks about a pattern called Array of Ancestors, which speeds up access to related data when joining documents.

> I believe the intrigue about these databases is infectious.

Concerns about navigating relationships are bound to the fact that in a relational database, repeating data is a sin. Databases are normalized to ensure this. When working in NoSQL databases, especially those that are distributed, denormalizing your data is useful and acceptable.

Querying and Updating

Each database comes with APIs for querying and updating. While they may not be part of the core API, a variety of language APIs are supplied through add-ons. As a .NET Framework entry into the document database world, RavenDB uses LINQ for querying—a nice benefit for .NET developers.

Other queries depend on predefined views and a pattern called map/reduce. The map part of this process uses the views, and the responsibility of the map differs between databases. The map also enables the database to distribute the query processing across multiple processors. Reduce takes the result of the map query (or queries, if it has been distributed) and aggregates the data into the results to be returned to the client.

Map/reduce is a pattern, and the various databases have their own implementations. Rob Ashton provides an interesting comparison of how RavenDB and CouchDB perform map/reduce at bit.ly/940CME.

While RavenDB requires predefined views for querying, and CouchDB only lets you query through map/reduce, MongoDB (also using views and map/reduce) additionally provides the ability to do ad hoc querying. For the most part, however, when moving away from the known schemas and relational nature of the SQL databases, the ability to perform ad hoc querying is one of the features you lose. By having tight control over the querying, the document databases are able to promise their fast performance.

A Database Revolution

There are *so* many nonrelational databases out there under the NoSQL umbrella. And now that the door has opened, it's inspiring more to come as folks look at what's available and dream of how they might improve on it. I think RavenDB is a great example of this, and you can watch how Rahien is evolving the database as he continues to dream about how to make it better or becomes inspired by users.

I believe the intrigue about these databases is infectious. I definitely look forward to digging further and learning more. But even the three I've looked at are so interesting that it's hard for this Libra to choose among them, because at present, I'm solving a curiosity problem and not a real business problem, and relational databases happen to be the right fit for my current projects. ■

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 the highly acclaimed book, "Programming Entity Framework" (O'Reilly Media, 2010). Follow her on Twitter at twitter.com/julielerman.*

THANKS to the following technical experts for reviewing this article: *Ted Neward and Savas Parastatidis*

Experience the Devexpress Difference

"As a training, mentoring and consulting company, we are often put on the spot as to which vendors we like for .Net tools. There is only one answer from my company and that is Developer Express. We have used Developer Express tools in our projects for the past five years and the company continues to impress me with the quality of their tools.

They simply work. You get what you pay for. Mileage will vary with other vendors but I can assure you Developer Express is a sure bet. "

–Mark Dunn, MCT, MCAD, MCDBA, MCSD .Net

"Our flagship product needed extensive visualizations including charts and graphs. We were looking for a single charting system that could address all of these needs, and handle large volumes of data. It needed to look attractive yet blend into our application.

With *XtraCharts* we were able to create high performance, real-time graphs of performance data through to detailed analytical bar charts. *XtraCharts* was the only option that was fast enough to handle the tens of thousands of data points our customers routinely throw at it. "

–Kendall Miller

Read More User Comments at:
DevExpress.com/Comments

Embedding RavenDB into an ASP.NET MVC 3 Application

Justin Schwartzenberger

Attention to the NoSQL movement is growing within the Microsoft .NET Framework community as we continue to hear of companies sharing their implementation experiences of it in applications that we know and use. With this heightened awareness comes the curiosity to dig in and identify how a NoSQL data store could provide benefits or other potential solutions to the software that developers are currently crafting. But where do you start, and how hard is the learning curve? Maybe an even more relevant concern: How much time and effort are required to fire up a new data storage solution and start writing code against it? After all, you have the setup process of SQL Server for a new application down to a science, right?

This article discusses:

- Running RavenDB in embedded mode
- RavenDB objects
- The Unit of Work pattern
- MVC implementations
- Searching objects

Technologies discussed:

RavenDB, ASP.NET MVC 3

Code download available at:

code.msdn.microsoft.com/mag201111RavenDB

Word has reached the .NET community on the wings of a raven about a new option for a NoSQL-type data-layer implementation. RavenDB (ravendb.net) is a document database designed for the .NET/Windows platform, packaged with everything you need to start working with a nonrelational data store. RavenDB stores documents as schema-less JSON. A RESTful API exists for direct interaction with the data store, but the real advantage lies within the .NET client API that comes bundled with the install. It implements the Unit of Work pattern and leverages LINQ syntax to work with documents and queries. If you've worked with an object-relational mapper (ORM)—such as the Entity Framework (EF) or NHibernate—or consumed a WCF Data Service, you'll feel right at home with the API architecture for working with documents in RavenDB.

The learning curve for getting up and running with an instance of RavenDB is short and sweet. In fact, the piece that may require the most planning is the licensing strategy (but even that's minimal). RavenDB offers an open source license for projects that are also open source, but a commercial license is required for closed source commercial projects. Details of the license and the pricing can be found at ravendb.net/licensing. The site states that free licensing is available for startup companies or those looking to use it in a noncommercial, closed source project. Either way, it's worthwhile to

quickly review the options to understand the long-term implementation potential before any prototyping or sandbox development.

RavenDB Embedded and MVC

RavenDB can be run in three different modes:

1. As a Windows service
2. As an IIS application
3. Embedded in a .NET application

The first two have a fairly simple setup process, but come with some implementation strategy overhead. The third option, embedded, is extremely easy to get up and running. In fact, there's a NuGet package available for it. A call to the following command in the Package Manager Console in Visual Studio 2010 (or a search for the term "ravendb" in the Manage NuGet Packages dialog) will deliver all of the references needed to start working with the embedded version of RavenDB:

```
Install-Package RavenDB-Embedded
```

Details of the package can be found on the NuGet gallery site at bit.ly/ns64W1.

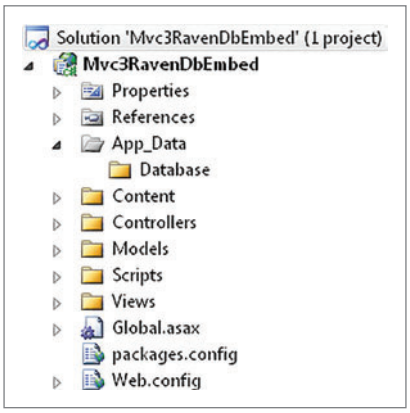


Figure 1 **App_Data Directory Structure**

> Word has reached the .NET community on the wings of a raven about a new option for a NoSQL-type data-layer implementation.

Adding the embedded version of RavenDB to an ASP.NET MVC 3 application is as simple as adding the package via NuGet and giving the data store files a directory location. Because ASP.NET applications have a known data directory in the framework named App_Data, and most hosting companies provide read/write access to that directory with little or no configuration required, it's a good place to store the data files. When RavenDB creates its file storage, it builds a handful of directories and files in the directory path provided to it. It won't create a top-level directory to store everything. Knowing that, it's worthwhile to add the ASP.NET folder named App_Data via the Project context menu in Visual Studio 2010 and then create a subdirectory in the App_Data directory for the RavenDB data (see **Figure 1**).

A document data store is schema-less by nature, hence there's no need to create an instance of a database or set up any tables. Once the first call to initialize the data store is made in code, the files required to maintain the data state will be created.

Working with the RavenDB Client API to interface with the data store requires an instance of an object that implements the Raven.Client.IDocumentStore interface to be created and initialized. The API has two classes, DocumentStore and EmbeddedDocumentStore, that implement the interface and can be used depending on the mode in which RavenDB is running. There should only be one instance per data store during the lifecycle of an application. I can create a class to manage a single connection to my document store that will let me access the instance of the IDocumentStore object via a static property and have a static method to initialize the instance (see **Figure 2**).

The static property getter checks a private static backing field for a null object and, if null, it throws an InvalidOperationException. I throw an exception here, rather than calling the Initialize method, to keep the code thread-safe. If the Instance property were allowed to make that call and the application relied upon referencing the property to do the initialization, then there would be a chance that more than one user could hit the application at the same time, resulting in simultaneous calls to the Initialize method. Within the Initialize method logic, I create a new instance of the Raven.Client.Embedded.EmbeddableDocumentStore and set the ConnectionStringName property to the name of a connection string that was added to the web.config file by the install of the RavenDB NuGet package. In the web.config, I set the value of the connection string to a syntax that RavenDB understands in order to configure it to use the embedded local version of the data store. I also map the file directory to the Database directory I created in the App_Data directory of the MVC project:

```
<connectionStrings>
  <add name="RavenDB " connectionString="DataDir = ~\App_Data\Database" />
</connectionStrings>
```

The IDocumentStore interface contains all of the methods for working with the data store. I return and store the EmbeddableDocumentStore object as an instance of the interface type IDocumentStore so I have the flexibility of changing the instantiation

Figure 2 **Class for DocumentStore**

```
public class DataDocumentStore
{
  private static IDocumentStore instance;

  public static IDocumentStore Instance
  {
    get
    {
      if(instance == null)
        throw new InvalidOperationException(
          "IDocumentStore has not been initialized.");
      return instance;
    }
  }

  public static IDocumentStore Initialize()
  {
    instance = new EmbeddableDocumentStore { ConnectionStringName = "RavenDB" };
    instance.Conventions.IdentityPartsSeparator = "-";
    instance.Initialize();
    return instance;
  }
}
```

Figure 3 **Bookmark Class**

```
public class Bookmark
{
  public string Id { get; set; }
  public string Title { get; set; }
  public string Url { get; set; }
  public string Description { get; set; }
  public List<string> Tags { get; set; }
  public DateTime DateCreated { get; set; }

  public Bookmark()
  {
    this.Tags = new List<string>();
  }
}
```

of the EmbeddedDocumentStore object to the server version (DocumentStore) if I want to move away from the embedded version. This way, all of my logic code that will handle my document object management will be decoupled from the knowledge of the mode in which RavenDB is running.

RavenDB will create document ID keys in a REST-like format by default. An "Item" object would get a key in the format "items/104." The object model name is converted to lowercase and is pluralized, and a unique tracking identity number is appended after a forward slash with each new document creation. This can be problematic in an MVC application, as the forward slash will cause a new route parameter to be parsed. The RavenDB Client API provides a way to change the forward slash by setting the IdentityPartsSeparator value. In my DataDocumentStore.Initialize method, I'm setting the IdentityPartsSeparator value to a dash before I call the Initialize method on the EmbeddableDocumentStore object, to avoid the routing issue.

Adding a call to the DataDocumentStore.Initialize static method from the Application_Start method in the Global.asax.cs file of my MVC application will establish the IDocumentStore instance at the first run of the application, which looks like this:

```
protected void Application_Start()
{
  AreaRegistration.RegisterAllAreas();
  RegisterGlobalFilters(GlobalFilters.Filters);
  RegisterRoutes(RouteTable.Routes);

  DataDocumentStore.Initialize();
}
```

The learning curve for getting up and running with an instance of RavenDB is short and sweet.

From here I can make use of the IDocumentStore object with a static call to the DataDocumentStore.Instance property to work on document objects from my embedded data store within my MVC application.

RavenDB Objects

To get a better understanding of RavenDB in action, I'll create a prototype application to store and manage bookmarks. RavenDB is designed to work with Plain Old CLR Objects (POCOs), so there's no need to add property attributes to guide serialization. Creating a class to represent a bookmark is pretty straightforward. **Figure 3** shows the Bookmark class.

RavenDB will serialize the object data into a JSON structure when it goes to store the document. The well-known "Id" named property will be used to handle the document ID key. RavenDB will create that value—provided the Id property is empty or null when making the call to create the new document—and will store it in a @metadata element for the document (which is used to handle the document key at the data-store level). When requesting a document, the RavenDB Client API code will set the document ID key to the Id property when it loads the document object.

Adding the embedded version of RavenDB to an ASP.NET MVC 3 application is as simple as adding the package via NuGet and giving the data store files a directory location.

The JSON serialization of a sample Bookmark document is represented in the following structure:

```
{
  "Title": "The RavenDB site",
  "Url": "http://www.ravendb.net",
  "Description": "A test bookmark",
  "Tags": ["mvc","ravendb"],
  "DateCreated": "2011-08-04T00:50:40.3207693Z"
}
```

The Bookmark class is primed to work well with the document store, but the Tags property is going to pose a challenge in the UI layer. I'd like to let the user enter a list of tags separated by commas in a single text box input field and have the MVC model binder map all of the data fields without any logic code seeping into my views or controller actions. I can tackle this by using a custom model binder for mapping a form field named "TagsAsString" to the Bookmark.Tags field. First, I create the custom model binder class (see **Figure 4**).

Figure 4 **BookmarkModelBinder.cs**

```
public class BookmarkModelBinder : DefaultModelBinder
{
  protected override void OnModelUpdated(ControllerContext controllerContext,
    ModelBindingContext bindingContext)
  {
    var form = controllerContext.HttpContext.Request.Form;
    var tagsAsString = form["TagsAsString"];
    var bookmark = bindingContext.Model as Bookmark;
    bookmark.Tags = string.IsNullOrEmpty(tagsAsString)
      ? new List<string>()
      : tagsAsString.Split(',').Select(i => i.Trim()).ToList();
  }
}
```

Then I update the Globals.asax.cs file to add the BookmarkModelBinder to the model binders at application startup:

```
protected void Application_Start()
{
  AreaRegistration.RegisterAllAreas();
  RegisterGlobalFilters(GlobalFilters.Filters);
  RegisterRoutes(RouteTable.Routes);

  ModelBinders.Binders.Add(typeof(Bookmark), new BookmarkModelBinder());
  DataDocumentStore.Initialize();
}
```

To handle populating an HTML text box with the current tags in the model, I'll add an extension method to convert a List<string> object to a comma-separated string:

```
public static string ToCommaSeparatedString(this List<string> list)
{
  return list == null ? string.Empty : string.Join(", ", list);
}
```

Unit of Work

The RavenDB Client API is based on the Unit of Work pattern. To work on documents from the document store, a new session needs to be opened; work needs to be done and saved; and the session needs to close. The session handles change tracking and operates in a manner that's similar to a data context in the EF. Here's an example of creating a new document:

```
using (var session = documentStore.OpenSession())
{
  session.Store(bookmark);
  session.SaveChanges();
}
```

It's optimal to have the session live throughout the HTTP request so it can track changes, use the first-level cache and so on. I'll create a base controller that will use the DocumentDataStore.Instance to open a new session on *action executing*, and on *action executed* will save changes and then dispose of the session object (see **Figure 5**). This allows me to do all of the work desired during the execution of my action code with a single open session instance.

MVC Controller and View Implementation

The BookmarksController actions will work directly with the IDocumentSession object from the base class and manage all of the Create, Read, Update and Delete (CRUD) operations for the documents. **Figure 6** shows the code for the bookmarks controller.

> A document data store is schema-less by nature, hence there's no need to create an instance of a database or set up any tables.

The IDocumentSession.Query<T> method in the Index action returns a result object that implements the IEnumerable interface, so I can use the OrderByDescending LINQ expression to sort the items and call the ToList method to capture the data to my return object. The IDocumentSession.Load method in the Details action takes in a document ID key value and de-serializes the matching document to an object of type Bookmark.

Figure 5 **BaseDocumentStoreController**

```
public class BaseDocumentStoreController : Controller
{
  public IDocumentSession DocumentSession { get; set; }

  protected override void OnActionExecuting(ActionExecutingContext filterContext)
  {
    if (filterContext.IsChildAction)
      return;
    this.DocumentSession = DataDocumentStore.Instance.OpenSession();
    base.OnActionExecuting(filterContext);
  }

  protected override void OnActionExecuted(ActionExecutedContext filterContext)
  {
    if (filterContext.IsChildAction)
      return;
    if (this.DocumentSession != null && filterContext.Exception == null)
      this.DocumentSession.SaveChanges();
    this.DocumentSession.Dispose();
    base.OnActionExecuted(filterContext);
  }
}
```

> RavenDB will create document ID keys in a REST-like format by default.

The Create method with the HttpPost verb attribute sets the CreateDate property on the bookmark item and calls the IDocumentSession.Store method off of the session object to add a new document record to the document store. The Update method with the HttpPost verb can call the IDocumentSession.Store method as well, because the Bookmark object will have the Id value already set. RavenDB will recognize that Id and update the existing document with the matching key instead of creating a new one. The DeleteConfirmed action calls a Delete method off of the IDocumentSession.Advanced.DatabaseCommands object, which provides a way to delete a document by key without having to load the object first. I don't need to call the IDocumentSession.SaveChanges method from within any of these actions, because I have the base controller making that call on *action executed*.

All of the views are pretty straightforward. They can be strongly typed to the Bookmark class in the Create, Edit and Delete markups, and to a list of bookmarks in the Index markup. Each view can directly reference the model properties for display and input fields. The one place where I'll need to vary on object property reference is with the input field for the tags. I'll use the ToCommaSeparatedString extension method in the Create and Edit views with the following code:

```
@Html.TextBox("TagsAsString", Model.Tags.ToCommaSeparatedString())
```

This will allow the user to input and edit the tags associated with the bookmark in a comma-delimited format within a single text box.

HTML

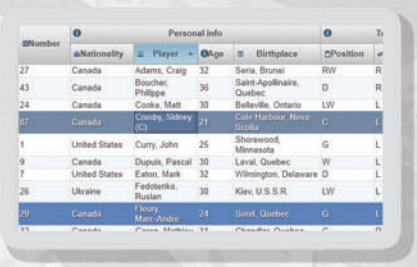
SEMANTICS CSS3 5 SVG AJAX jQuery

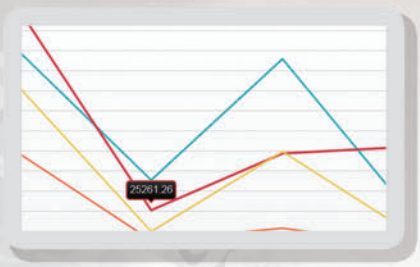
MEET THE NEW WEB STACK

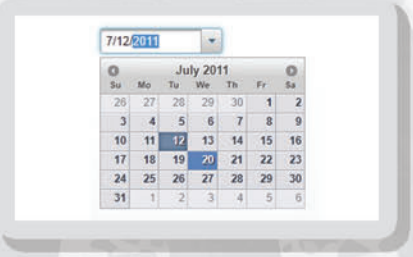
all new

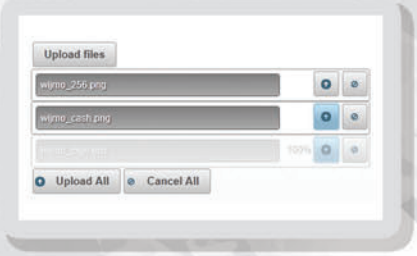
From WebForms to MVC

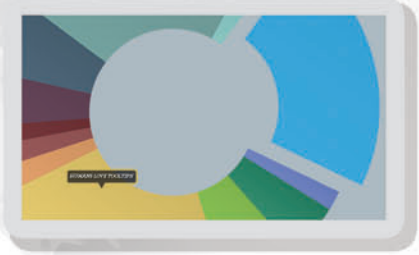
Our new web stack is the ultimate kit for web development. We have tools that range from WebForms to MVC and from pure client-side to robust server-side development all powered by our core technology: Wijmo.











TOOLS FOR

WEBFORMS | MVC | JAVASCRIPT

ComponentOne® Studio® for ASP.NET wijmo®

Full-featured ASP.NET Server-side Controls
plus ASP.NET Ajax Extender Controls

DOWNLOAD YOUR FREE TRIALS @
componentone.com/webstack

C1 ComponentOne®

Searching Objects

With all of my CRUD operations in place, I can turn my attention to adding one last bit of functionality: the ability to filter the bookmark list by tags. In addition to implementing the IEnumerable interface, the return object from the IDocumentSession.Query method also implements the IOrderedQueryable and IQueryable interfaces from the .NET Framework. This allows me to use LINQ to filter and sort my queries. For example, here's a query of the bookmarks created in the past five days:

```
var bookmarks = session.Query<Bookmark>()
  .Where( i=> i.DateCreated >= DateTime.UtcNow.AddDays(-5))
  .OrderByDescending(i => i.DateCreated)
  .ToList();
```

Here's one to page through the full list of bookmarks:

```
var bookmarks = session.Query<Bookmark>()
  .OrderByDescending(i => i.DateCreated)
  .Skip(pageCount * (pageNumber - 1))
  .Take(pageCount)
  .ToList();
```

RavenDB will build dynamic indexes based on the execution of these queries that will persist for "some amount of time" before being disposed of. When a similar query is rerun with the same parameter structure, the temporary dynamic index will be used. If the index is used enough within a given period, the index will be made permanent. These will persist beyond the application lifecycle.

I can add the following action method to my BookmarksController class to handle getting bookmarks by tag:

```
public ViewResult Tag(string tag)
{
  var model = new BookmarksByTagViewModel { Tag = tag };
  model.Bookmarks = this.DocumentSession.Query<Bookmark>()
    .Where(i => i.Tags.Any(t => t == tag))
    .OrderByDescending(i => i.DateCreated)
    .ToList();
  return View(model);
}
```

I expect this action to be hit on a regular basis by users of my application. If that's indeed the case, this dynamic query will get turned into a permanent index by RavenDB with no additional work needed on my part.

> With the emergence of RavenDB, the .NET community appears to finally have a NoSQL document store-type solution catered toward it.

A Raven Sent to Awaken Us

With the emergence of RavenDB, the .NET community appears to finally have a NoSQL document store-type solution catered toward it, allowing Microsoft-centric shops and developers to glide through the nonrelational world that so many other frameworks and languages have been navigating for the past few years. Nevermore shall we hear the cries of a lack of nonrelational love for the Microsoft stack. RavenDB is making it easy for .NET developers to start playing and prototyping with a nonrelational data store by bundling the install with a clean client API that mimics data-management techniques that developers are already employing. While the perennial argument between relational and nonrelational surely won't die out, the ease of trying out something "new" should help lead to a better understanding of how and where a nonrelational solution can fit within an application architecture. ■

Figure 6 **BookmarksController Class**

```
public class BookmarksController : BaseDocumentStoreController
{
  public ViewResult Index()
  {
    var model = this.DocumentSession.Query<Bookmark>()
      .OrderByDescending(i => i.DateCreated)
      .ToList();
    return View(model);
  }

  public ViewResult Details(string id)
  {
    var model = this.DocumentSession.Load<Bookmark>(id);
    return View(model);
  }

  public ActionResult Create()
  {
    var model = new Bookmark();
    return View(model);
  }

  [HttpPost]
  public ActionResult Create(Bookmark bookmark)
  {
    bookmark.DateCreated = DateTime.UtcNow;
    this.DocumentSession.Store(bookmark);
    return RedirectToAction("Index");
  }

  public ActionResult Edit(string id)
  {
    var model = this.DocumentSession.Load<Bookmark>(id);
    return View(model);
  }

  [HttpPost]
  public ActionResult Edit(Bookmark bookmark)
  {
    this.DocumentSession.Store(bookmark);
    return RedirectToAction("Index");
  }

  public ActionResult Delete(string id)
  {
    var model = this.DocumentSession.Load<Bookmark>(id);
    return View(model);
  }

  [HttpPost, ActionName("Delete")]
  public ActionResult DeleteConfirmed(string id)
  {
    this.DocumentSession.Advanced.DatabaseCommands.Delete(id, null);
    return RedirectToAction("Index");
  }
}
```

JUSTIN SCHWARTZENBERGER, *CTO at DealerHosts, has been entrenched in Web application development for quite a while, traversing the syntactic jungles of PHP, classic ASP, Visual Basic, VB.NET and ASP.NET Web Forms. As an early adopter of ASP.NET MVC in 2007, he decided to refactor his Web stack focus to all things MVC. He contributes articles, speaks at user groups, maintains a blog at iwantmymvc.com and can be followed on Twitter at twitter.com/schwarty.*

THANKS to the following technical expert for reviewing this article: *Ayende Rahien*

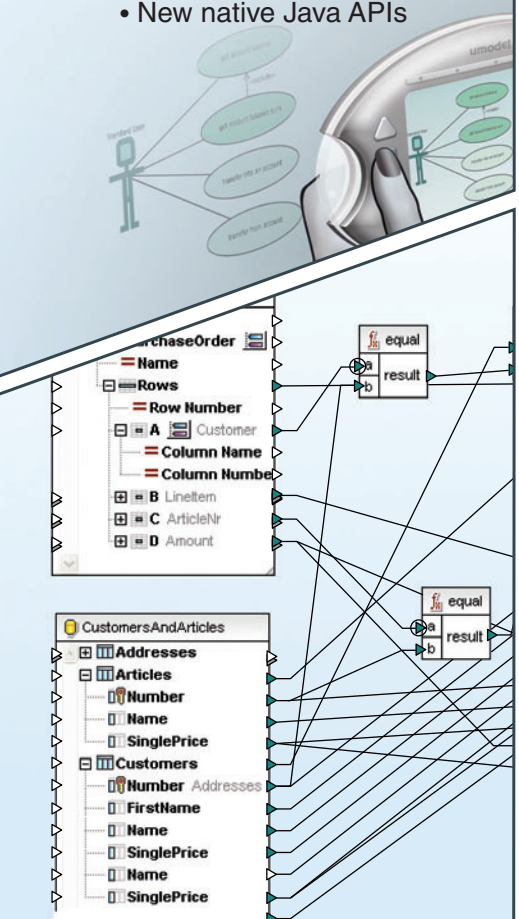
Visualize software works of art with the complete set of tools from Altova®

ALTOVA® missionkit®

The MissionKit® is an integrated suite of UML, XML, and data integration tools for today's software architect.

NEW in Version 2012:

- Support for Model Driven Architecture: transform UML models between C#, VB, Java, databases, & XSD
- Intelligent HTML5 editing
- JDBC database driver support
- New native Java APIs



The Altova MissionKit includes multiple tools for software architects:

UModel® – UML tool for software modeling

- Support for all UML diagram types, MDA, SQL database diagrams, BPMN, SysML
- Reverse engineering and code generation in Java, C#, VB.NET

XMLSpy® – XML editor & development environment

- Support for all XML-based technologies
- Royalty-free Java, C#, C++ code generation

MapForce® – graphical data mapping tool

- Mapping between XML, databases, EDI, flat files, XBRL, Excel 2007+, Web services
- Royalty-free Java, C#, C++ code generation

Plus up to five additional tools…

Download a 30 day free trial!

Try before you buy with a free, fully functional trial from **www.altova.com**

Scan to learn more about these Altova MissionKit tools.

ComponentArt

Mobile Dashboards

ComponentArt's unique technology delivers rich BI dashboards to any tablet, smartphone, PC or Mac, while being perfectly positioned for Windows 8.

Try them live at ComponentArt.com

ComponentArt

Your Data in a Whole New Light

Building a 'Mango' App

Andrew Whitechapel

"Mango" is the internal code name for the Windows Phone SDK 7.1 release, and of course the name of a delicious tropical fruit. There are many ways you can use mangoes—for example, in pies, salads and a range of cocktails. The mango is also said to provide a number of health benefits, and it has an interesting cultural history. In this article, I'll examine Mangolicious, a Windows Phone SDK 7.1 application about mangoes. The application provides a range of mango recipes, cocktails and facts, but the real purpose is to explore some of the big new features in the 7.1 release, specifically:

- Local database and LINQ to SQL
- Secondary tiles and deep linking
- Silverlight/XNA integration

The application's user experience is simple: The main page offers a panorama, with a menu on the first panorama item, a dynamic selection of the current season's recipes and cocktails on the second item, and some simple "about" information on the third item, as shown in **Figure 1**.

Both the menu and the items in the Seasonal Highlights section act as links to navigate to the other pages in the application. The majority of the pages are straightforward Silverlight pages, and one page is dedicated to an integrated XNA game. Here's a summary of the tasks required to build this application, from start to finish:

1. Create the basic solution in Visual Studio.
2. Independently create the database for the recipe, cocktail and fact data.
3. Update the application to consume the database and expose it for data binding.
4. Create the various UI pages and data bind them.
5. Set up the Secondary Tiles feature to allow the user to pin Recipe items to the phone's Start page.
6. Incorporate an XNA game into the application.

This article discusses:

- Creating the solution
- Building the database and DataContext class
- Consuming the database
- Generating the UI
- Using tiles
- Adding an XNA game

Technologies discussed:

Silverlight, XNA, Windows Phone SDK 7.1

Code download available at:

code.msdn.microsoft.com/mag201111MangoApp

Create the Solution

For this application, I'll use the Windows Phone Silverlight and XNA Application template in Visual Studio. This generates a solution with three projects; after renaming, these are summarized in **Figure 2**.

Create the Database and DataContext Class

The Windows Phone SDK 7.1 release introduces support for local databases. That is, an application can store data in a local database file (SDF) on the phone. The recommended approach is to create the database in code, either as part of the application itself or via a separate helper application that you build purely to create the

Figure 1 **Mangolicious Panorama Main Page**

database. It makes sense to create the database within your application in scenarios where you'll be creating most or all of the data only when the application runs. For the Mangolicious application, I have only static data, and I can populate the database in advance.

To do this, I'll create a separate database-creator helper application, starting with the simple Windows Phone Application template. To create the database in code, I need a class derived from DataContext, which is defined in the custom Phone version of the System.Data.Linq assembly. This same DataContext class can be used both in the helper application that creates the database and the main application that consumes the database. In the helper application, I must specify the database location to be in isolated storage, because that's the only location I can write to from a phone application. The class also contains a set of Table fields for each database table:

```
public class MangoDataContext : DataContext
{
  public MangoDataContext()
    : base("Data Source=isostore:/Mangolicious.sdf") { }

  public Table<Recipe> Recipes;
  public Table<Fact> Facts;
  public Table<Cocktail> Cocktails;
}
```

Figure 2 **Projects in a Windows Phone Silverlight and XNA Solution**

Project	Description
MangoApp	Contains the phone application itself, with a default MainPage and a secondary GamePage.
GameLibrary	An essentially empty project that has all the right references, but no code. Crucially, it includes a Content Reference to the Content project.
GameContent	An empty Content project, which will hold all the game assets (images, sound files and so on).

There's a 1:1 mapping between Table classes in the code and tables in the database. The Column properties map to the columns in the table in the database, and include the database schema properties such as the data type and size (INT, NVARCHAR and so on), whether the column may be null, whether it's a key column and so on. I define Table classes for all the other tables in the database in the same way, as shown in **Figure 3**.

Still, in the helper application—and using a standard Model-View-ViewModel (MVVM) approach—I now need a ViewModel class to mediate between the View (the UI) and the Model (the data) using the DataContext class. The ViewModel has a DataContext field and a set of collections for the table data (Recipes, Facts and Cocktails). The data is static, so simple List<T> collections are sufficient here. For the same reason, I only need *get* property accessors, not *set* modifiers (see **Figure 4**).

I also expose a public method—which I can invoke from the UI—to actually create the database and all the data. In this method,

Figure 3 **Defining Table Classes**

```
[Table]
public class Recipe
{
  private int id;
  [Column(
    IsPrimaryKey = true, IsDbGenerated = true,
    DbType = "INT NOT NULL Identity", CanBeNull = false,
    AutoSync = AutoSync.OnInsert)]
  public int ID
  {
    get { return id; }
    set
    {
      if (id != value)
      {
        id = value;
      }
    }
  }

  private string name;
  [Column(DbType = "NVARCHAR(32)")]
  public string Name
  {
    get { return name; }
    set
    {
      if (name != value)
      {
        name = value;
      }
    }
  }

    ... additional column definitions omitted for brevity
}
```

Figure 4 **Defining Collection Properties for Table Data in the ViewModel**

```
public class MainViewModel
{
  private MangoDataContext mangoDb;

  private List<Recipe> recipes;
  public List<Recipe> Recipes
  {
    get
    {
      if (recipes == null)
      {
        recipes = new List<Recipe>();
      }
    return recipes;
    }
  }

    ... additional table collections omitted for brevity
}
```

I create the database itself if it doesn't already exist and then create each table in turn, populating each one with static data. For example, to create the Recipe table, I create multiple instances of the Recipe class, corresponding to rows in the table; add all the rows in the collection to the DataContext; and finally commit the data to the database. The same pattern is used for the Facts and Cocktails tables (see **Figure 5**).

At a suitable point in the helper application—perhaps in a button click handler—I can then invoke this CreateDatabase method. When I run the helper (either in the emulator or on a physical device), the database file will be created in the application's isolated storage. The final task is to extract that file to the desktop so I can use it in the main application. To do this, I'll use the Isolated Storage Explorer tool, a command-line tool that ships with the Windows Phone SDK 7.1. Here's the command to take a snapshot of isolated storage from the emulator to the desktop:

```
"C:\Program Files\Microsoft SDKs\Windows Phone\v7.1\Tools\
IsolatedStorageExplorerTool\ISETool" ts xd {e0e7e3d7-c24b-498e-b88d-
d7c2d4077a3b} C:\Temp\IsoDump
```

This command assumes the tool is installed in a standard location. The parameters are explained in **Figure 6**.

Having extracted the SDF file to the desktop, I'm now finished with the helper application and can turn my attention to the Mangolicious application that will consume this database.

Figure 5 **Creating the Database**

```
public void CreateDatabase()
{
  mangoDb = new MangoDataContext();
  if (!mangoDb.DatabaseExists())
  {
    mangoDb.CreateDatabase();
    CreateRecipes();
    CreateFacts();
    CreateCocktails();
  }
}

private void CreateRecipes()
{
  Recipes.Add(new Recipe
  {
    ID = 1,
    Name = "key mango pie",
    Photo = "Images/Recipes/MangoPie.jpg",
    Ingredients = "2 cans sweetened condensed milk, ¾ cup fresh key lime
juice, ¼ cup mango purée, 2 eggs, ¾ cup chopped mango.",
    Instructions = "Mix graham cracker crumbs, sugar and butter until
well distributed. Press into a 9-inch pie pan. Bake for 20 minutes.
Make filling by whisking condensed milk, lime juice, mango purée and
egg together until blended well. Stir in fresh mango. Pour filling into
cooled crust and bake for 15 minutes.",
    Season = "summer"
  });

    ... additional Recipe instances omitted for brevity

  mangoDb.Recipes.InsertAllOnSubmit<Recipe>(Recipes);
  mangoDb.SubmitChanges();
}
```

Consume the Database

In the Mangolicious application, I add the SDF file to the project and also add the same custom DataContext class to the solution, with a couple of minor changes. In Mangolicious, I don't need to write to the database, so I can use it directly from the application install folder. Thus the connection string is slightly different from the one in the helper application. Also, Mangolicious defines a SeasonalHighlights table in code. There's no corresponding SeasonalHighlight table in the database. Instead, this code table pulls data from two underlying database tables (Recipes and Cocktails) and is used to populate the Seasonal Highlights panorama item. These two changes are the only differences in the DataContext class between the database-creation helper application and the Mangolicious database-consuming application:

```
public class MangoDataContext : DataContext
{
  public MangoDataContext()
    : base("Data Source=appdata:/Mangolicious.sdf;File Mode=read only;") { }

  public Table<Recipe> Recipes;
  public Table<Fact> Facts;
  public Table<Cocktail> Cocktails;
  public Table<SeasonalHighlight> SeasonalHighlights;
}
```

The Mangolicious application also needs a ViewModel class, and I can use the ViewModel class from the helper application as a starting point. I need the DataContext field and the set of List<T> collection properties for the data tables. On top of that, I'll add a string property to record the current season, computed in the constructor:

```
public MainViewModel()
{
  season = String.Empty;
  int currentMonth = DateTime.Now.Month;
  if (currentMonth >= 3 && currentMonth <= 5) season = "spring";
  else if (currentMonth >= 6 && currentMonth <= 8) season = "summer";
  else if (currentMonth >= 9 && currentMonth <= 11) season = "autumn";
  else if (currentMonth == 12 || currentMonth == 1 || currentMonth == 2)
    season = "winter";
}
```

Figure 6 **Isolated Storage Explorer Command-Line Parameters**

Parameter	Description
ts	"Take snapshot" (the command to download from isolated storage to the desktop).
xd	Short for XDE (that is, the emulator).
{e0e7e3d7-c24b-498e-b88d-d7c2d4077a3b}	The ProductID for the helper application. This is listed in the WMAppManifest.xml and is different for each application.
C:\Temp\IsoDump	Any valid path on the desktop where you want to copy the snapshot to.

Less Pain, More Gain

check out infragistics.com/reporting

NetAdvantage® for Reporting

EXPORT TO EXCEL, WORD AND PDF

Export reports from the client and server side in the popular format of your choice!

DATA ACCESS SUPPORT

Create MVVM-friendly reports with data accessed from an SQL Server, Oracle or any Object Data Source.

DESIGN INTERFACE

Optimize your data presentation and build attractive reports with an integrated and easy-to-use design-time experience.

REPORT VIEWER

View pixel-perfect reports with vector graphics in our Silverlight, ASP.NET, WPF and Windows Forms report viewer.

INFRAGISTICS
DESIGN / DEVELOP / EXPERIENCE

Infragistics Sales 800 231 8588 • Infragistics Europe Sales +44 (0) 800 298 9055 • Infragistics India +91 80 4151 8042 • @infragistics

Figure 7 **Loading Data at Startup**

```
public void LoadData()
{
  mangoDb = new MangoDataContext();
  if (!mangoDb.DatabaseExists())
  {
    mangoDb.CreateDatabase();
  }

  var seasonalRecipes = from r in mangoDb.Recipes
                        where r.Season == season
                        select new { r.ID, r.Name, r.Photo };
  var seasonalCocktails = from c in mangoDb.Cocktails
                          where c.Season == season
                          select new { c.ID, c.Name, c.Photo };

  seasonalHighlights = new List<SeasonalHighlight>();
  foreach (var v in seasonalRecipes)
  {
    seasonalHighlights.Add(new SeasonalHighlight {
      ID = v.ID, Name = v.Name, Photo = v.Photo, SourceTable="Recipes" });
  }
  foreach (var v in seasonalCocktails)
  {
    seasonalHighlights.Add(new SeasonalHighlight {
      ID = v.ID, Name = v.Name, Photo = v.Photo, SourceTable = "Cocktails" });
  }

  isDataLoaded = true;
}
```

The critical method in the ViewModel is the LoadData method. Here, I initialize the database and perform LINQ-to-SQL queries to load the data via the DataContext into my in-memory collections. I could preload all three tables at this point, but I want to optimize startup performance by delaying the loading of data unless and until the relevant page is actually visited. The only data I *must* load at startup is the data for the SeasonalHighlight table, because this is displayed on the main page. For this, I have two queries to select only rows from the Recipes and Cocktails tables that match the current season, and add the combined row sets to the collection, as shown in **Figure 7**.

I can use similar LINQ-to-SQL queries to build separate LoadFacts, LoadRecipes and LoadCocktails methods that can be used after startup to load their respective data on demand.

Create the UI

The main page consists of a Panorama with three PanoramaItems. The first item consists of a ListBox that offers a main menu for the application. When the user selects one of the ListBox items, I navigate to the corresponding page—that is, the collection page for either Recipes, Facts and Cocktails—or the Game page. Just before navigating, I make sure to load the corresponding data into the Recipes, Facts or Cocktails collections:

Figure 8 **Selecting from the Seasonal Highlights List**

```
SeasonalHighlight selectedItem =
  (SeasonalHighlight)SeasonalList.SelectedItem;
String navigationString = String.Empty;
if (selectedItem.SourceTable == "Recipes")
{
  App.ViewModel.LoadRecipes();
  navigationString =
    String.Format("/RecipePage.xaml?ID={0}", selectedItem.ID);
}
else if (selectedItem.SourceTable == "Cocktails")
{
  App.ViewModel.LoadCocktails();
  navigationString =
    String.Format("/CocktailPage.xaml?ID={0}", selectedItem.ID);
}
NavigationService.Navigate(
  new System.Uri(navigationString, UriKind.Relative));
```

Figure 9 **Fun Facts, One of the Collection List Pages**

Figure 10 **A Recipe Page with Pin Button**

```
switch (CategoryList.SelectedIndex)
{
  case 0:
    App.ViewModel.LoadRecipes();
    NavigationService.Navigate(
      new Uri("/RecipesPage.xaml", UriKind.Relative));
    break;

... additional cases omitted for brevity
}
```

When the user selects an item from the Seasonal Highlights list in the UI, I examine the selected item to see whether it's a Recipe or a Cocktail, and then navigate to the individual Recipe or Cocktail page, passing in the item ID as part of the navigation query string, as shown in **Figure 8**.

> Implicit styling is another new feature introduced in the Windows Phone SDK 7.1 as part of Silverlight 4.

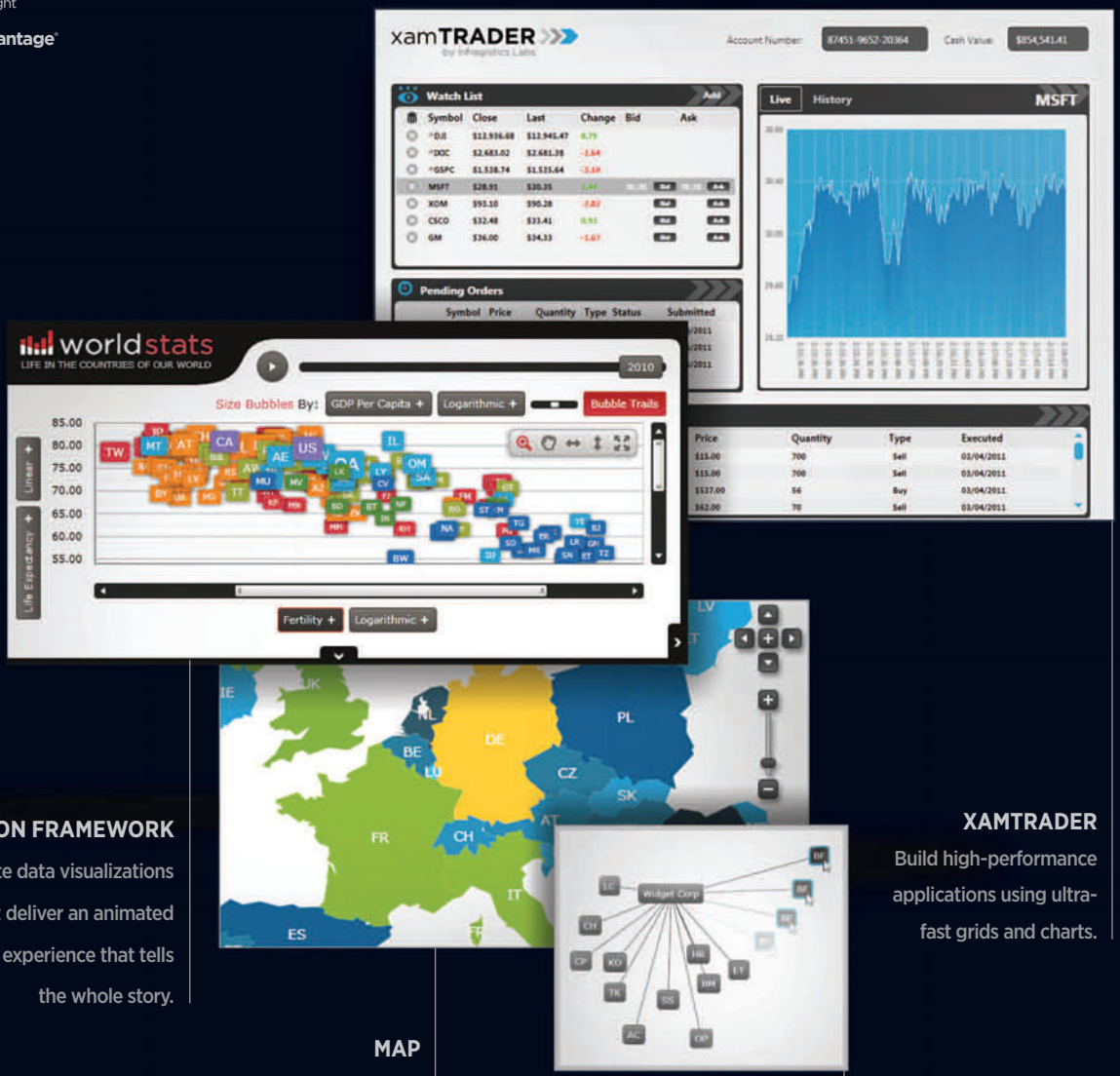
The user can navigate from the menu on the main page to one of three listing pages. Each of these pages data binds to one of the collections in the ViewModel to display a list of items: Recipes, Facts or Cocktails. Each of these pages offers a simple ListBox where each item in the list contains an Image control for the photo and a TextBlock for the name of the item. For example, **Figure 9** shows the FactsPage.

When the user selects an individual item from the Recipes, Facts or Cocktails lists, I navigate to the individual Recipe, Fact or Cocktail page, passing down the ID of the individual item in the

XAML-IFY YOUR APPS

check out infragistics.com/xaml

- **NetAdvantage®** for Silverlight Data Visualization
- **NetAdvantage®** for WPF Data Visualization
- **NetAdvantage®** for Silverlight
- **NetAdvantage®** for WPF



MOTION FRAMEWORK

Create data visualizations that deliver an animated user experience that tells the whole story.

MAP

Ensure your geospatial data really goes places with a feature-laden, interactive Map Control for your applications.

XAMTRADER

Build high-performance applications using ultra-fast grids and charts.

NETWORK NODE

Help your users make the connection with visual representations of simple or complex network relationships.

INFRAGISTICS
DESIGN / DEVELOP / EXPERIENCE

Infragistics Sales 800 231 8588 • Infragistics Europe Sales +44 (0) 800 298 9055 • Infragistics India +91 80 4151 8042 • @infragistics

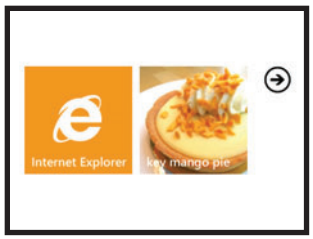


Figure 11 **Pinned Recipe Tile (Front)**

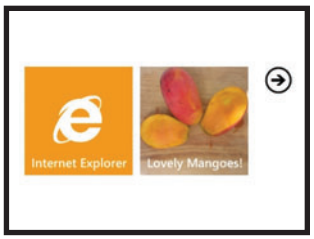


Figure 12 **Pinned Recipe Tile (Back)**

navigation query string. Again, these pages are almost identical across the three types, each one offering an Image and some text below. Note that I don't define an explicit style for the databound TextBlocks, but that they all nonetheless use TextWrapping=Wrap. This is done by declaring a TextBlock style in the App.xaml.cs:

```
<Style TargetType="TextBlock" BasedOn="{StaticResource
  PhoneTextNormalStyle}">
  <Setter Property="TextWrapping" Value="Wrap"/>
</Style>
```

The effect of this is that any TextBlock in the solution that doesn't explicitly define its own style will implicitly use this one instead. Implicit styling is another new feature introduced in the Windows Phone SDK 7.1 as part of Silverlight 4.

> Although a "home" mechanism isn't impossible to achieve, it's behavior that you should weigh carefully before introducing.

The codebehind for each of these pages is simple. In the OnNavigatedTo override, I extract the individual item ID from the query string, find that item from the ViewModel collection and data bind to it. The code for the RecipePage is a little more complex than the others—the additional code in this page is all related to the HyperlinkButton positioned at the top-right-hand corner of the page. This can be seen in **Figure 10**.

Secondary Tiles

When the user clicks the "pin" HyperlinkButton on the individual Recipe page, I pin that item as a tile on the phone's Start page. The act of pinning takes the user to the Start page and deactivates the application. When a tile is pinned in this way, it animates periodically, flipping between front and back, as shown in **Figure 11** and **Figure 12**.

Subsequently, the user may tap this pinned tile, which navigates directly to that item within the application. When he reaches the page, the "pin" button will now have an "unpin" image. If he unpins the page, it will be removed from the Start page, and the application continues.

Here's how this works. In the OnNavigatedTo override for the RecipePage, after doing the standard work to determine which specific Recipe to data bind to, I formulate a string that I can use later as the URI for this page:

```
thisPageUri = String.Format("/RecipePage.xaml?ID={0}", recipeID);
```

In the button click handler for the "pin" button, I first check to see if a tile for this page already exists, and if it doesn't, I create it now. I create the tile using the current Recipe data: the image and the name. I also set a single static image—and static text—for the back of the tile. At the same time, I take the opportunity to repaint the button itself, using the "unpin" image. If, on the other hand, the tile did already exist, then I must be in the click handler because the user has chosen to unpin the tile. In this case, I delete the tile and repaint the button using the "pin" image, as shown in **Figure 13**.

Note that if the user taps the pinned tile to get to the Recipe page, and then presses the phone's hardware Back button, he will exit out of the application altogether. This is potentially confusing because the user normally expects that he will only exit the application when he presses Back from the main page, not from any other page. However, the alternative would be to provide some kind of "home" button on the Recipe page to allow the user to navigate back to the rest of the application. Unfortunately, this would also be confusing, because when the user gets to the main page and presses Back, he'd be back on the pinned Recipe page, instead of exiting out of the application. For this reason, although a "home" mechanism isn't impossible to achieve, it's behavior that you should weigh carefully before introducing.

Incorporate an XNA Game

Recall that I originally created the application as a Windows Phone Silverlight and XNA Application solution. This gave me three projects. I've been working with the main MangoApp project to build out the non-game functionality. The GameLibrary project acts as a "bridge" between the Silverlight MangoApp and the XNA GameContent. It's referenced in the MangoApp project, and in turn references the

Figure 13 **Pinning and Unpinning Pages**

```
private void PinUnpin_Click(object sender, RoutedEventArgs e)
{
  tile = ShellTile.ActiveTiles.FirstOrDefault(
    x => x.NavigationUri.ToString().Contains(thisPageUri));
  if (tile == null)
  {
    StandardTileData tileData = new StandardTileData
    {
      BackgroundImage = new Uri(
        thisRecipe.Photo, UriKind.RelativeOrAbsolute),
      Title = thisRecipe.Name,
      BackTitle = "Lovely Mangoes!",
      BackBackgroundImage =
        new Uri("Images/BackTile.png", UriKind.Relative)
    };

    ImageBrush brush = (ImageBrush)PinUnpin.Background;
    brush.ImageSource =
      new BitmapImage(new Uri("Images/Unpin.png", UriKind.Relative));
    PinUnpin.Background = brush;
    ShellTile.Create(
      new Uri(thisPageUri, UriKind.Relative), tileData);
  }
  else
  {
    tile.Delete();
    ImageBrush brush = (ImageBrush)PinUnpin.Background;
    brush.ImageSource =
      new BitmapImage(new Uri("Images/Pin.png", UriKind.Relative));
    PinUnpin.Background = brush;
  }
}
```

Deliver the Ultimate User Experience

check out infragistics.com/ultimate

NetAdvantage® ULTIMATE

FINANCIAL CHARTING

With support for multiple chart styles, and technical indicators built in, financial charting capabilities are on the money.

TREEMAP

Communicate the relative differences in data weight more effectively, with customizable color and flexible layouts.

OLAP AXIS CHART

Take your data to new depths with the seemingly endless drilldown capability of the OLAP Axis Chart.

OLAP GRID

Provide highly-interactive pivot grid functionality in all of your applications.

INFRAGISTICS
DESIGN / DEVELOP / EXPERIENCE

Infragistics Sales 800 231 8588 • Infragistics Europe Sales +44 (0) 800 298 9055 • Infragistics India +91 80 4151 8042 • @infragistics

GameContent project. This needs no further work. There are two main tasks required in order to incorporate a game into my phone application:

- Enhance the GamePage class in the MangoApp project to include all the game logic.
- Enhance the GameContent project to provide images and sounds for the game (no code changes).

Looking briefly at the enhancements that Visual Studio generated for a project that integrates Silverlight and XNA, the first thing to note is that the App.xaml declares a SharedGraphicsDeviceManager. This manages the sharing of the screen between the Silverlight and XNA runtimes. This object is also the sole reason for the additional AppServiceProvider class in the project. This class is used to cache the shared graphics device manager so it's available to anything that needs it in the app, both Silverlight and XNA. The App class has an AppServiceProvider field, and it also exposes some additional properties for XNA integration: a ContentManager and a GameTimer. These are all initialized in the new InitializeXnaApplication method, along with a GameTimer, which is used to pump the XNA message queue.

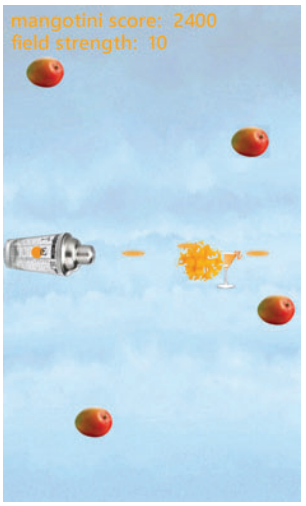


Figure 14 **The XNA Game in Progress**

The interesting work is how to integrate an XNA game.

The interesting work is how to integrate an XNA game within a Silverlight phone application. The game itself is actually less interesting. So, for this exercise, rather than spending effort writing a complete game from scratch, I'll adapt an existing game—specifically, the XNA game tutorial on AppHub: bit.ly/h0ZM4o.

In my adaptation, I have a cocktail shaker, represented by the Player class in code, that fires projectiles at incoming mangoes (enemies). When I hit a mango, it breaks open and morphs into a mangotini. Every mango hit adds 100 to the score. Every time the cocktail shaker collides with a mango, player field strength is reduced by 10. When the field strength goes to zero, the game ends. The user can also end the game at any time by pressing the phone's Back button, as expected. **Figure 14** shows the game in progress.

I don't need to make any changes to the (nearly empty) GamePage.xaml. Rather, all the work is done in the codebehind. Visual Studio generates starter code for this GamePage class, as described in **Figure 15**.

The game is a direct adaptation of the AppHub tutorial, which contains two projects: the Shooter game project and the ShooterContent content project. The content contains images and sound files. Although it doesn't affect the application code, I can change these to align them with the mango theme of my application, and that's just a question of replacing PNG and WAV files. The required (code) changes are all in the Shooter game project. Guidelines for migrating from the Game Class to Silverlight/XNA are on AppHub: bit.ly/iHl3jz.

First, I must copy the Shooter game project files into my existing MangoApp project. Also, I copy the ShooterContent content files into my existing GameContent project. **Figure 16** summarizes the existing classes in the Shooter game project.

In order to incorporate this game into my phone application, I need to make the following changes to the GamePage class:

- Copy all fields from the Game1 class to the GamePage class. Also copy the field initialization in the Game1.Initialize method to the GamePage constructor.
- Copy the LoadContent method, and all methods to add and update enemies, projectiles and explosions. None of these require any changes.

Figure 15 **GamePage Starter Code**

Field/Method	Purpose	Changes Required
ContentManager	Loads and manages the lifetime of content from the content pipeline.	Add code to use this to load images and sounds.
GameTimer	In the XNA game model, the game performs actions when Update and Draw events are fired, and these events are governed by a timer.	Unchanged.
SpriteBatch	Used to draw textures in XNA.	Add code to use this in the Draw method to draw the game objects (player, enemies, projectiles, explosions and so on).
GamePage Constructor	Creates a timer and hooks up its Update and Draw events to OnUpdate and OnDraw methods.	Keep the timer code, and additionally initialize the game objects.
OnNavigatedTo	Sets up graphics sharing between Silverlight and XNA, and starts the timer.	Keep the sharing and timer code, and additionally load content into the game, including any previous state from isolated storage.
OnNavigatedFrom	Stops the timer and turns off XNA graphics sharing.	Keep the timer and sharing code, and additionally store the game score and player health to isolated storage.
OnUpdate	(Empty), handles the GameTimer.Update event.	Add code to calculate game object changes (player position, number and position of enemies, projectiles and explosions).
OnDraw	(Empty), handles the GameTimer.Draw event.	Add code to draw game objects, game score and player health.

Powerful Tools for Developers

v4.5!

High-Performance PDF Printer Driver

CITRIX ready

- Create accurate PDF documents in a fraction of the time needed with other tools
- WHQL tested for all Windows 32 and 64-bit platforms
- Produce fully compliant PDF/A documents
- Standard PDF features included with a number of unique features
- Interface with any .NET or ActiveX programming language

v4.5!

PDF Editor for .NET, now Webform Enabled

- Edit, process and print PDF 1.7 documents programmatically
- Fast and lightweight 32 and 64-bit managed code assemblies for Windows, WPF and Web applications
- Support for dynamic objects such as edit-fields and sticky-notes
- Save image files directly to PDF, with optional OCR
- Multiple image compression formats such as PNG, JBIG2 and TIFF

New!

PDF Integration into Silverlight Applications

- Server-side PDF component based on the robust Amyuni PDF Creator ActiveX or .NET components
- Client-side C# Silverlight 3 control provided with source-code
- Optimization of PDF documents prior to converting them into XAML
- Conversion of PDF edit-boxes into Silverlight TextBox objects
- Support for other document formats such as TIFF and XPS

OCR Module available with royalty-free licensing!

The new OCR Module from Amyuni enables developers to:

- Convert non-searchable PDF files into searchable PDFs
- Create searchable PDF documents out of various image formats such as multi-page TIFF, JPEG or PNG while applying text recognition
- Compress image based PDF documents using high compression JBIG2 or more standard CCITT, JPEG and PNG compression formats

The Amyuni OCR module is based on the Tesseract Library with the Amyuni PDF technology being used to process and create the PDF documents.

Learn more at www.amyuni.com

Windows 7

CERTIFIED FOR Windows Server 2008

More Development Tools Available at:

www.amyuni.com

USA and Canada
Toll Free: 1 866 926 9864
Support: (514) 868 9227
Info: sales@amyuni.com

Europe
Sales: (+33) 1 30 61 07 97
Support: (+33) 1 30 61 07 98
Customizations: management@amyuni.com

AMYUNI Technologies

- Extract all uses of the GraphicsDeviceManager to use a GraphicsDevice property instead.
- Extract the code in the Game1.Update and Draw methods into the GamePage.OnUpdate and OnDraw timer event handlers.

A conventional XNA game creates a new GraphicsDeviceManager, whereas in a phone application I'll already have a SharedGraphicsDeviceManager that exposes a GraphicsDevice property, and that's all I really need. To simplify things, I'll cache a reference to the GraphicsDevice as a field in my GamePage class.

In a standard XNA game, the Update and Draw methods are overrides of virtual methods in the base Microsoft.Xna.Framework.Game class. However, in an integrated Silverlight/XNA application, the GamePage class isn't derived from the XNA Game class, so I must abstract the code from Update and Draw and insert it into the OnUpdate and OnDraw event handlers instead. Note that some of the game object classes (such as Animation, Enemy and Player), the Update and Draw methods, and some of the helper methods called by Update take a GameTime parameter. This is defined in Microsoft.Xna.Framework.Game.dll, and it should generally be considered a bug if a Silverlight application contains any reference to this assembly. The GameTime parameter can be completely replaced by the two Timespan properties—TotalTime and ElapsedTime—exposed from the GameTimerEventArgs object that's passed in to the OnUpdate and OnDraw timer event handlers. Apart from GameTime, I can port the Draw code unchanged.

The original Update method tests GamePad state and conditionally calls Game.Exit. This isn't used in an integrated Silverlight/XNA application, so it must not be ported to the new method:

```
//if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed)
//{
//    // this.Exit();
//}
```

The new Update method is now little more than a harness that calls into other methods to update the various game objects. I update the parallaxing background even when the game is over, but only update the player, enemies, collision, projectiles and explosions if the player is still alive. These helper methods calculate the number and positions of the various game objects. Having eliminated the use of GameTime, these can all be ported over unchanged—with one exception:

```
private void OnUpdate(object sender, GameTimerEventArgs e)
{
  backgroundLayer1.Update();
  backgroundLayer2.Update();

  if (isPlayerAlive)
  {
    UpdatePlayer(e.TotalTime, e.ElapsedTime);
    UpdateEnemies(e.TotalTime, e.ElapsedTime);
    UpdateCollision();
    UpdateProjectiles();
    UpdateExplosions(e.TotalTime, e.ElapsedTime);
  }
}
```

The UpdatePlayer method *does* need a little tweaking. In the original version of the game, when the player's health dropped to zero, it was reset to 100, which meant that the game looped forever. In my adaptation, when player health drops to zero, I set a flag to false. I test for this flag in both the OnUpdate method and OnDraw. In OnUpdate, the flag value determines whether or not to compute further changes to the objects; in OnDraw, it determines whether to draw the objects or to draw a "game over" screen with the final score:

```
private void UpdatePlayer(TimeSpan totalTime, TimeSpan elapsedTime)
{
...unchanged code omitted for brevity.

  if (player.Health <= 0)
  {
    //player.Health = 100;
    //score = 0;
    gameOverSound.Play();
    isPlayerAlive = false;
  }
}
```

Play Along

In this article, I've looked at how to develop applications against several of the new features in Windows Phone SDK 7.1: local databases, LINQ to SQL, secondary tiles and deep linking, and Silverlight/XNA integration. The 7.1 release offers many more new features and enhancements to existing features. For further details, see the following links:

- What's New in the Windows Phone SDK: bit.ly/c2RmNr
- Tiles: bit.ly/oQlu15
- Combining Silverlight and XNA: bit.ly/p4RncQ
- Local Database Overview for Windows Phone: bit.ly/l23UQM

The final version of the Mangolicious application is available on the Windows Phone Marketplace at bit.ly/nuJcTA (note: Zune software is needed for access). Note that the sample uses the Silverlight for Windows Phone Toolkit (a free download, available at bit.ly/qiHnTT). ■

Figure 16 **Shooter Game Classes**

Class	Purpose	Changes Required
Animation	Animates the various sprites in my game: the player, enemy objects, projectiles and explosions.	Eliminate GameTime.
Enemy	A sprite that represents the enemy objects that the user shoots. These will be mangoes in my adaptation.	Eliminate GameTime.
Game1	The controlling class for the game.	Merge into the GamePage class.
ParallaxingBackground	Animates the cloud background images to provide a 3D parallax effect.	None.
Player	A sprite that represents the user's character in the game. This will be a cocktail shaker in my adaptation.	Eliminate GameTime.
Program	Only used if the game targets Windows or Xbox.	Unused; can be deleted.
Projectile	A sprite that represents the projectiles that the player fires at the enemies.	None.

ANDREW WHITECHAPEL *has been a developer for more than 20 years and currently works as a program manager on the Windows Phone team, responsible for core pieces of the application platform.*

THANKS to the following technical experts for reviewing this article: *Nick Gravelyn, Brian Hudson and Himadri Sarkar*

Hey Developers
Are you working with FILES?

Aspose provides premium file APIs for most business-centric formats.
.NET, Java, SharePoint, SQL Reporting & JasperReports

Aspose.Words

DOC, DOCX, RTF, HTML, PDF, XPS & other document formats.

Aspose.Cells

XLS, XLSX, XLSM, XLTX, CSV SpreadsheetML & image formats.

Aspose.Slides

PPT, PDF, PPS, POTX, PPTX & other formats.

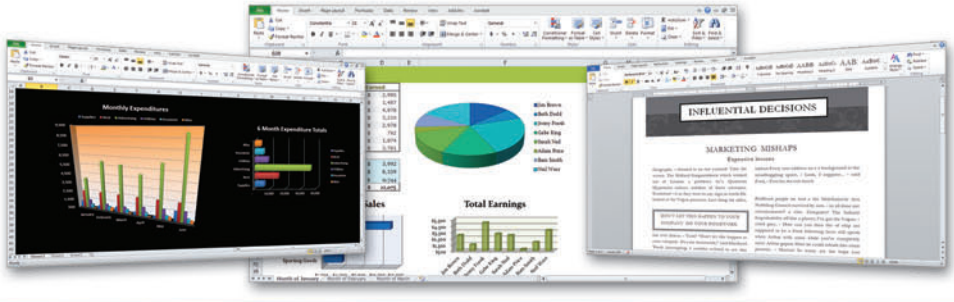
Aspose.Pdf Aspose.BarCode Aspose.Tasks
Aspose.Email Aspose.Diagram Aspose.OCR

Features:

- Create
- Edit
- Convert
- Import
- Export
- Render
- Save
- Print

When you buy Aspose.Total, you will receive our complete suite and any future products.

100% Standalone - No Office Automation



ASPOSE
Your File Format Experts

Get your FREE evaluation copy at **http://www.aspose.com**.

US Sales: 1.888.277.6734 · sales@aspose.com · EU Sales: +44 (0)800 098 8425 · sales.europe@aspose.com

Enterprise Sales – enterprise.sales@aspose.com

Deploying LightSwitch Applications to Windows Azure

Mike Wade

The new Microsoft Visual Studio LightSwitch aims to simplify the creation of classic line-of-business (LOB) forms-over-data applications. LightSwitch reduces the overhead of building these applications by performing much of the heavy lifting of creating connections to databases (the data-storage tier), displaying a professional UI (the presentation tier) and implementing business logic code (the logic tier).

To further simplify your life, you can host such applications on Windows Azure, an Internet-scale cloud services platform hosted on Microsoft datacenters. The platform includes Windows Azure, a cloud services OS, and SQL Azure, a database service hosted in the cloud. Hosting a LightSwitch application on the Windows Azure platform eliminates the need to dedicate resources to infrastructure management, such as Web servers and data servers: Windows Azure can take care of all of that for you.

In this article I'll take a look at how to deploy a LightSwitch application to Windows Azure using the Vision Clinic sample application, which is available for download at bit.ly/LightSwitchSamples. Vision Clinic is a simple LOB application designed for an optometrist's office. The application can be used to manage patients and appointments, as well as products the clinic's patients may require. It uses two databases: the intrinsic application database, which manages patients and their appointments, and an attached database called PrescriptionContoso that manages products available for sale at the clinic. The original walk-through shows how to deploy the application as a two-tier desktop application: the application runs completely on the client user's machine, but the data for the application is hosted elsewhere. After completing the steps in this article you'll be able to publish your application on Windows Azure.

This article discusses:

- Attaching a LightSwitch application to a SQL Azure Data Source
- Hosting the application on Windows Azure
- The output files resulting from publishing to Windows Azure
- Using additional Windows Azure features

Technologies discussed:

Visual Studio LightSwitch, Windows Azure

Attaching to a SQL Azure Data Source

The walk-through makes use of a sample external database—the PrescriptionContoso database, which you can download from MSDN and install to your local development machine. Because the PrescriptionContoso database is an attached data source, you'll have to create and populate the database yourself. When the finished application is deployed to Windows Azure, it will use both an attached data source and the intrinsic data source hosted in the cloud. Let's get the hard part out of the way first: creating that attached data source in SQL Azure.

Start by logging in to your Windows Azure account at windows.azure.com. If you don't already have an account, you can sign up for one at the same site. Once you've signed in to the Windows Azure Platform Management Portal, select the Database node in

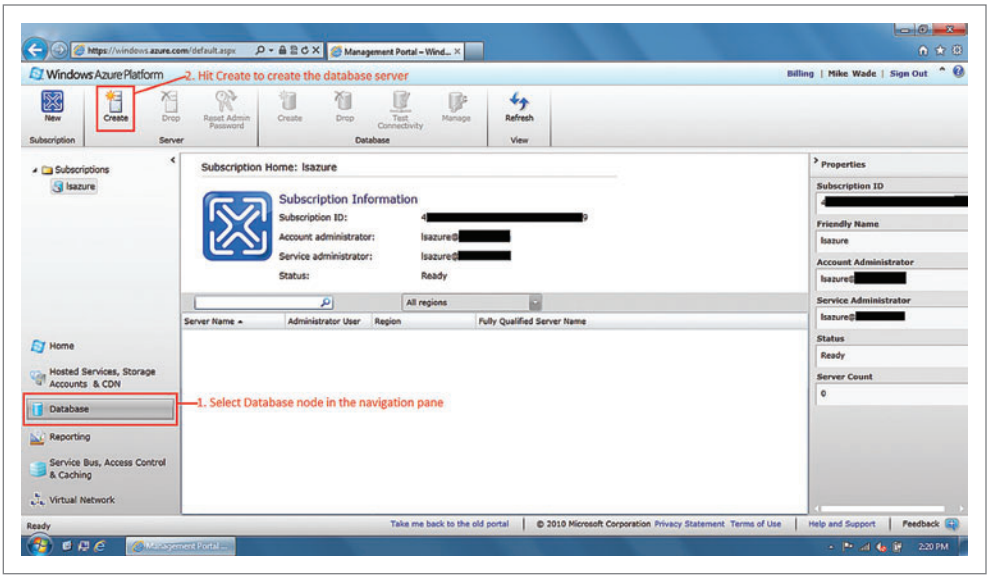


Figure 1 **Windows Azure Platform Management Portal: Creating a Database Server**

With the database server created, you can now view its properties. Select the database server in the navigation pane (mine was called "pq96r63lrm"). The items list includes all of the firewall rules you created, as well as a default "master" database.

You'll want to create a new database for the Prescription-Contoso information. Make sure your database server has been selected in the navigation pane, and then click the Create button from the ribbon in the Windows Azure Platform Management Portal. For your database name, type "PrescriptionContoso" and leave the default settings for Edition and Maximum size (see **Figure 3**).

the navigation pane on the left and view the database information for your subscription in the central items list. (See bit.ly/pcwCLX for a description of the layout of the portal.) If your account doesn't yet have a database subscription, select your subscription in the navigation panel, and choose Create from the ribbon across the top (see **Figure 1**). This will launch a server-creation wizard, allowing you to choose your SQL Server hosting region and your administrator login name and password, and to create firewall rules (see **Figure 2**). You'll need to add at least two firewall rules for your server:

1. Select the checkbox to allow other Windows Azure services to access this server. This enables the Vision Clinic application, which will eventually be hosted on Windows Azure, to access this database.
2. Add a range of IP addresses that will also be able to access the server. These are IP addresses of computers that can manage the database through the Windows Azure Platform Management Portal. For example, the IP address of my PC is 131.107.xxx.yyy, so I added a range of addresses starting at 131.107.0.0 and ending at 131.107.0.255. Don't forget: you'll still need your Windows Azure subscription login information to manage the database through the portal, but setting this firewall rule allows Visual Studio to access to the database during development. Once you've created and deployed the application, you can remove this firewall rule to prevent any external machines from accessing the database.

Now it's time to add the tables to the PrescriptionContoso database. Select the new database in the left-hand navigation panel and click the Manage button. This will open a new Web page that allows you to manage the PrescriptionContoso database. On the new page, select New Table. This will open a table editor page (see **Figure 4**). Name the table "Product" and enter the information shown in **Figure 5** for the schema.

Create another new table, name this one "ProductRebate," and enter the information in **Figure 6** for the schema. As with all attached data sources, the connection string LightSwitch uses to interact with the PrescriptionContoso

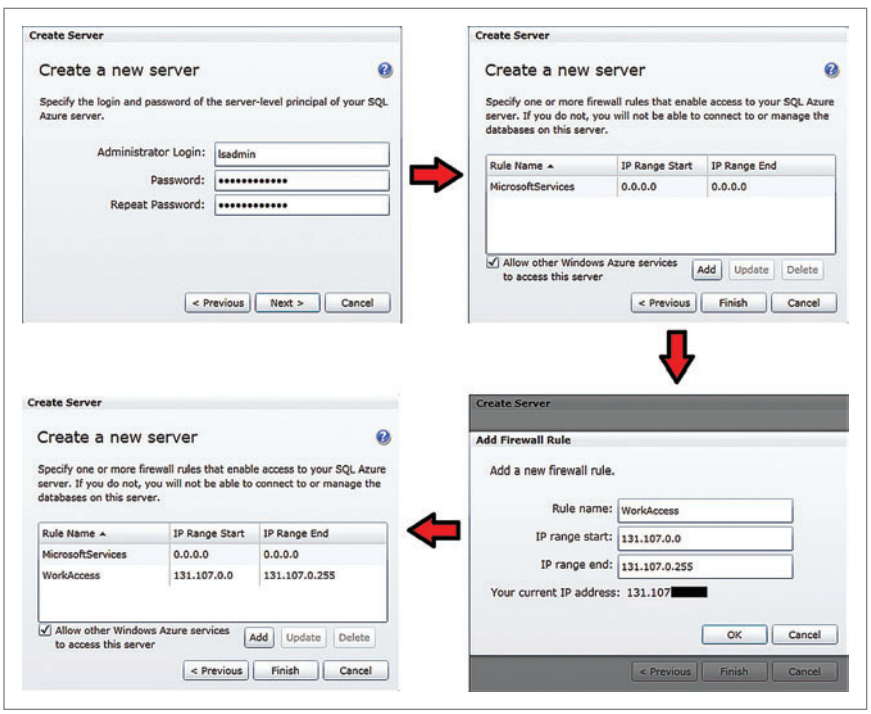


Figure 2 **Using the Wizard to Create a Database Server**

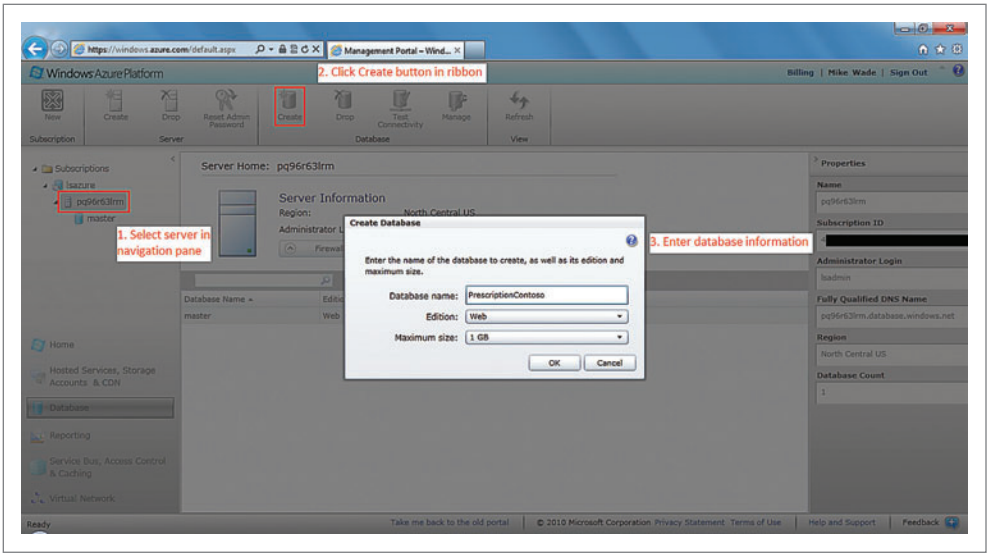


Figure 3 **Creating the PrescriptionContoso Database**

database will be stored in the web.config file generated as project output. Three-tier applications store the web.config file on the IIS server where it can't be seen by application users. However, with two-tier applications, all users who install the application can see the web.config file. Thus, it's important to create a non-admin login and user for the PrescriptionContoso database for use at run time. This user will not be a full database server administrator and will be able to perform only Create, Read, Update, Delete (CRUD) operations on the PrescriptionContoso database. You create a new login for the database by opening a connection to your master database and executing the following commands in SQL Server Management Studio Express or using sqlcmd (see bit.ly/ok2Mdh for more information):

```
CREATE LOGIN <user> WITH password='<password>';
```

Be sure to substitute a new <user> and <password>. The password you provide must be strong (see bit.ly/p4BEwU).

Now connect to the PrescriptionContoso database with your SQL Azure administrator account and create a user:

```
CREATE USER <user> FROM LOGIN <user>;
```

The database schema has been created, but the table contains no data. You can add data to the table using the data designer on the Windows Azure Platform Management Portal or by migrating the data from the PrescriptionContoso.mdf file included in the sample to the Windows Azure database. You can also use the bcp utility that comes with Microsoft SQL Server 2008 and Microsoft SQL Server 2008 Express (bit.ly/bhH7Ub). Yet another option is to add the data through a running application. It's possible to have a two-tier application attach to a data source hosted on SQL Azure, so let's populate the database by publishing and running the Vision Clinic example as a two-tier application.

With the schema in place on Windows Azure for the attached data source, it's time to publish the application. You should be able to follow the deployment steps in the Vision Clinic walk-through (bit.ly/py9yna) until step 8. On the Other Connection Information page, enter the connection string to the database that was just created. You'll find the connection string in the Windows Azure Platform Management Portal by selecting the database in the navigation pane and hitting the Connection Strings button in the properties pane. Copy the ADO.Net connection string (see **Figure 7**) and paste it into the PrescriptionContoso textbox in the LightSwitch publish wizard. Open the Connection Properties dialog and make sure you type the user name and password for the new user you created. Use the Test Connection button to make sure you're able to connect to the SQL Azure database.

After you enter the connection string for the attached data source, you can finish the publish wizard and wait for your application to be published. Once it's published, you can view the generated web.config file among the publish output and see that the connection string to your Windows Azure database is now in place. Because the application is still two-tiered, end users will need the Microsoft .NET Framework 4 installed on their computers, but the application can now read and write data to the PrescriptionContoso database on SQL Azure.

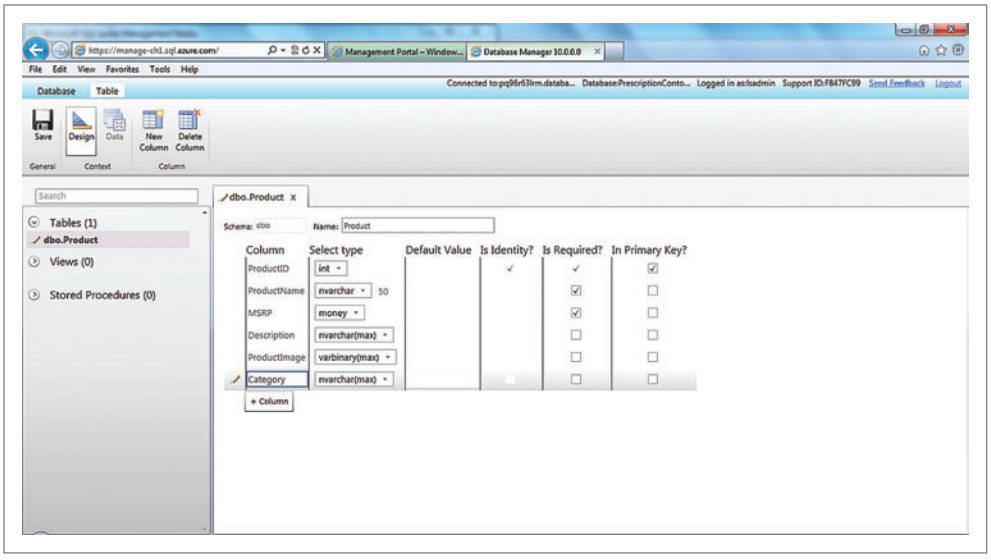


Figure 4 **Adding the Product Table to the PrescriptionContoso Database**

5 YEARS OF EXCELLENCE

XCEED
DataGrid
for WPF

Mature, feature-packed, and lightning-fast. The most adopted and trusted WPF datagrid.

IBM®
U2 SystemBuilder™

"IBM U2 researched existing third-party solutions and identified Xceed DataGrid for WPF as the most suitable tool. The datagrid's performance and solid foundation take true advantage of WPF and provide the extensibility required for IBM U2 customers to take their applications to the next level in presentation design and styling."

Vincent Smith
U2 Tools Product Manager at IBM

Microsoft®
Visual Studio® Team System 2010

"Using Xceed DataGrid for WPF in Microsoft Visual Studio System 2010 helped us greatly reduce the time and resources necessary for developing all the data presentation feature we needed. Working with Xceed has been a pleasure."

Norman Guadagno
Director of Product Marketing
for Microsoft Visual Studio Team System

XCEED Professional Themes for WPF

Theme your entire app in minutes. Flawless styles for all official WPF controls.

XCEED DataGrid for Silverlight

Incredible streaming technology. Speed up your app and say goodbye to paging.

XCEED Ultimate ListBox for WPF

NEW

The world's first streaming listbox. Simple, drop-in upgrade to the WPF listbox.

XCEED Ultimate ListBox for Silverlight

NEW

Fast and fluid, with ground-breaking streaming technology.

Visual Studio PARTNER

XCEED MULTI-TALENTED COMPONENTS

Figure 5 **Schema for the Product Table**

Column	Type	Default Value	Is Identity?	Is Required?	In Primary Key?
ProductID	Int		Yes	Yes	Yes
ProductName	nvarchar (50)			Yes	
MSRP	Money			Yes	
Description	nvarchar(max)				
ProductImage	varbinary(max)				
Category	nvarchar(max)				

Figure 6 **Schema for the ProductRebate Table**

Column	Type	Default Value	Is Identity?	Is Required?	In Primary Key?
ProductRebateID	Int		Yes	Yes	Yes
ProductID	Int			Yes	
RebateStart	Smalldatetime				
RebateEnd	Smalldatetime				
Rebate	Money				

Hosting the Application on Windows Azure

Next, let's take a look at hosting the application completely on Windows Azure. Because Windows Azure is running IIS, it's possible to have the same types of applications on Windows Azure as on an IIS server residing in your enterprise. Application users will only need to have Microsoft Silverlight 4 or higher to run the application, rather than the full .NET Framework 4, greatly simplifying client deployments. You'll modify the Vision Clinic sample to publish the application to Windows Azure. The service tier will still require the .NET Framework 4, but Windows Azure will automatically provision that for you.

Before you can publish Vision Clinic to Windows Azure, there's some additional preparation that needs to be done to your subscription. You'll need to pre-create a hosted service in the Windows Azure Platform Management Portal. This service is what will run the server-side code of the deployed application. You'll also need to create a storage account, which will be used to store the application binaries while the application is deploying. The application data will be stored in a SQL Azure database.

Navigate back to the portal and select the Home button in the navigation pane (see **Figure 8**). In the ribbon, you should see a button marked New Hosted Service. Clicking this button shows a new window in which to create your Hosted Service. Enter a service name, URL prefix and region for your application. For Deployment Options, choose "Do not deploy"; the LightSwitch publish wizard will be deploying the application. The Storage Account is created in a similar fashion: Click the New Storage Account button on the ribbon and fill out the fields in the pop-up that comes up. You should now be able to navigate to the Hosted Services and Storage in the Windows Azure Platform Management Portal. The rest

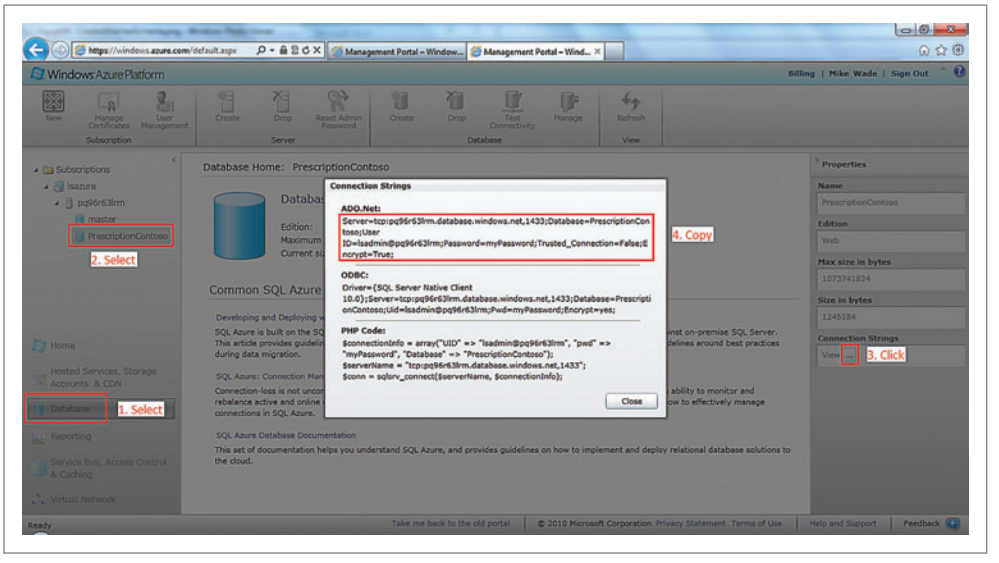


Figure 7 **Getting the Connection String for the PrescriptionContoso Database**

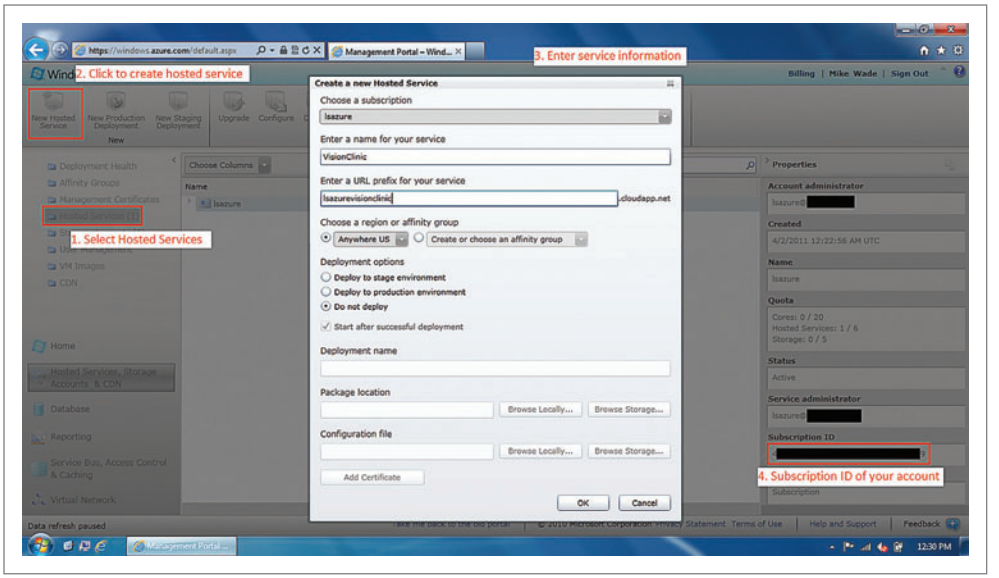


Figure 8 **Creating a Hosted Service in the Windows Azure Platform Management Portal**

Meet the EXPERTS

James Johnson
Product Manager

ACTIVEREPORTS
GrapeCity PowerTools

WE ARE REPORTING

- ✓ ActiveReports the award-winning .NET product allows Microsoft Visual Studio developers to provide royalty-free reporting solutions.
- ✓ ActiveReports now supports Silverlight and Azure.
- ✓ ActiveReports features a fast and flexible reporting engine.
- ✓ ActiveReports has an event-driven API to completely control the rendering of reports; Windows Forms, Silverlight, Web viewer controls and an end user designer control.
- ✓ ActiveReports has a wide range of export formats including Excel, TIFF, and PDF.
- ✓ ActiveReports multi-language support, extensive customization, and XCopy deployment simplifies Windows and Web reporting.

GrapeCity PowerTools

www.GCPowerTools.com GvTv.GCPowerTools.com

Your Online Source for Developer News!

Welcome to GrapeVine TV!

GvTv.GCPowerTools.com

GrapeCity PowerTools

GCPowerTools.com

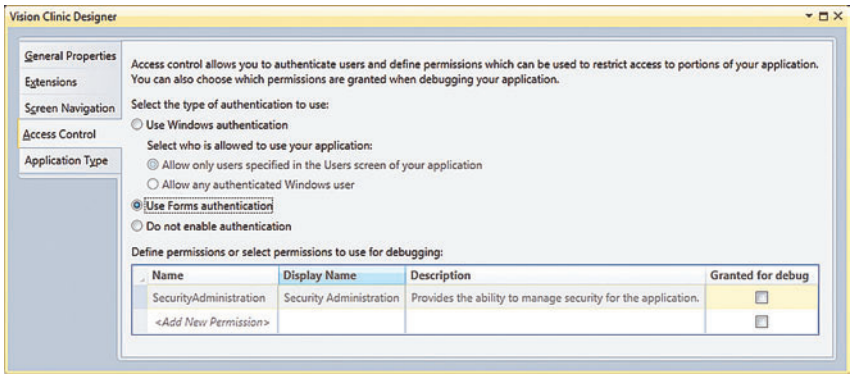


Figure 9 **Using Forms Authentication**

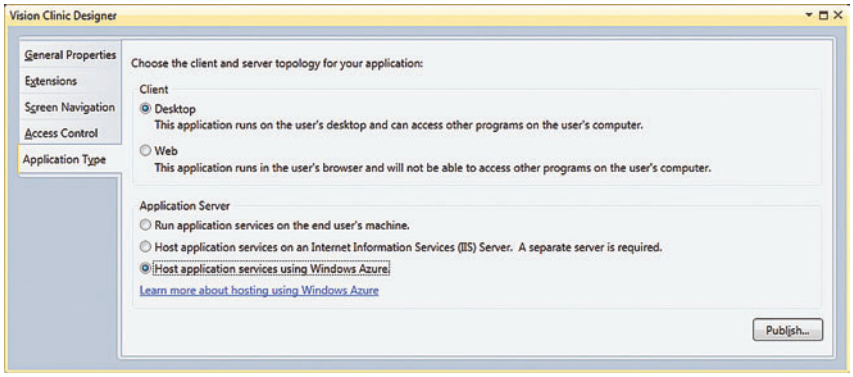


Figure 10 **Choosing to Host Application Services on Windows Azure**

of the Windows Azure configuration can be done through the LightSwitch publish wizard.

Let's make some updates to the Vision Clinic sample in the LightSwitch IDE. Because the application is being hosted in the cloud, it's a good idea to secure your LightSwitch application using Forms authentication. (For more information, see the article, "Securing Access to LightSwitch Applications," in this issue.) This is configured in the application properties access control tab of the VisionClinic application (see **Figure 9**). Now navigate to the Application Type tab. The application will remain a desktop application, because you want the application to be able to interact with other applications installed on an end user's computer (for example, exporting data to Microsoft Excel). For the Application Server Configuration, select Windows Azure (see **Figure 10**). Click on the Publish… button in the designer to begin the publishing process.

Hit Next twice in the wizard to get to the Connect to Windows Azure page. This page requires two pieces of information (see **Figure 11**):

1. The ID of your account subscription
2. A management certificate thumbprint

The subscription ID is obtained from the Properties pane of the hosted service in the Windows Azure Platform Management Portal. If this is the first time you're publishing to Windows Azure, you'll also have to create a new certificate that will be used to confirm your identity to the Windows Azure service during the publish process because there's no login during the publish process. In the drop-down, select the option to <Create new self-signed certificate…>. Doing this adds a new certificate to your computer's certificate store. Now you need to upload certificate information to your Windows Azure account. Choose Copy Path in the publish wizard and then go back to the portal and select the Management Certificates node in the navigation pane. Select Add Certificate from the ribbon and then choose the Browse… button from the ensuing dialog (see **Figure 12**). Paste in the path that you copied from the publish wizard—and your certificate should now be added.

The Azure Service Configuration page (see **Figure 13**) in the wizard allows you to specify the names of the hosted service and storage accounts that were created through the Windows Azure Platform Management Portal. These drop-downs should be automatically populated by the wizard once a valid subscription ID and management certificate have been entered on the previous page. You also can choose which Environment the application will be deployed to: Production or Staging. For this example, select Production.

Next, you'll need to specify the Security Settings for the application: a certificate that will be used to establish an SSL connection for the deployed application. Communicating with Windows Azure over HTTPS protects business data exchanged between client and server, as well as the user name and password when using Forms authentication information. The wizard allows you to select from existing certificates that have already been uploaded to Windows Azure, or to upload a new certificate, as shown in **Figure 14**. The drop-down lets you create a new self-signed certificate that can

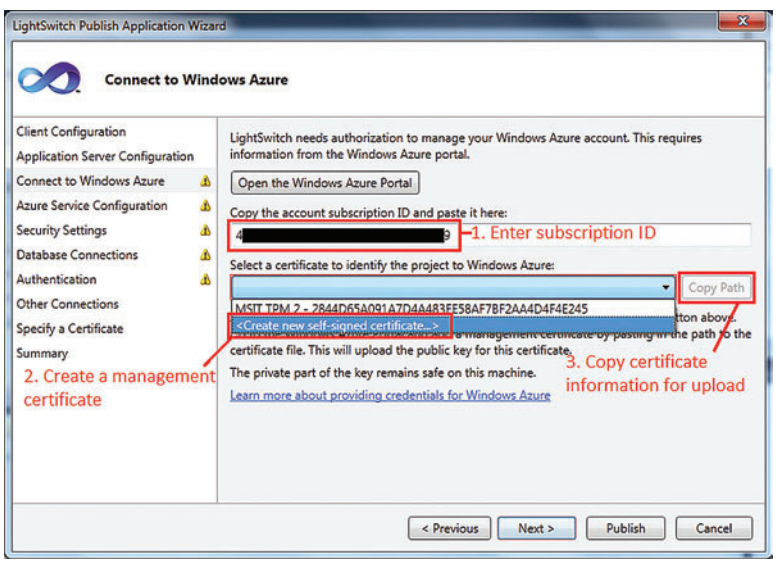


Figure 11 **Connecting to Windows Azure in the LightSwitch Publish Wizard**

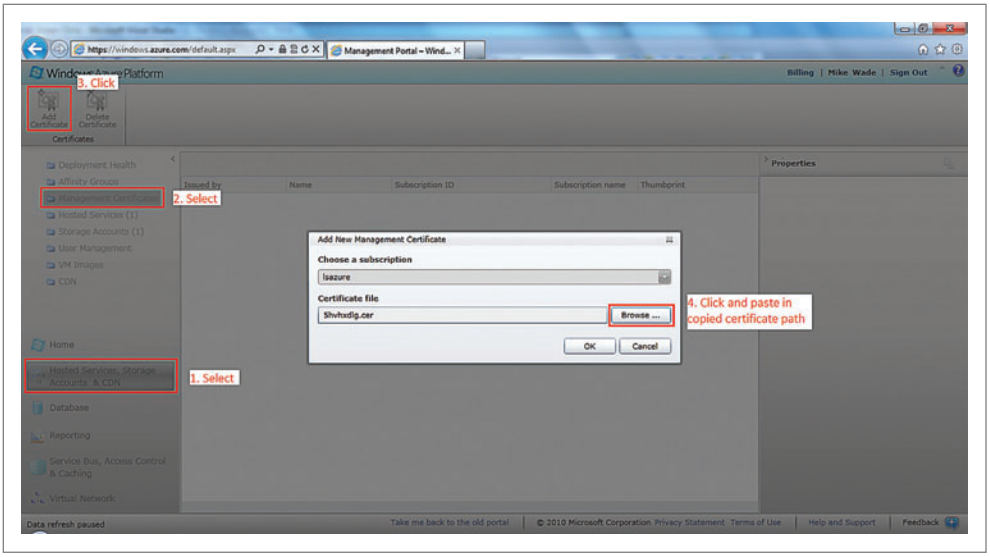


Figure 12 **Adding a Management Certificate to Windows Azure**

be uploaded to Windows Azure. The Silverlight out-of-browser launcher tool that LightSwitch desktop applications use requires a trusted certificate on the server when run over HTTPS. If the certificate isn't trusted, the application won't run correctly. The best approach is to use a certificate signed by a trusted Certificate Authority. A less-secure alternative is to pre-install the new self-signed certificate in the client's certificate store (see bit.ly/ra3CKG for more information). A LightSwitch client that runs in the browser can work with a self-signed certificate, but the browser will show a warning before loading the Silverlight application.

The next piece of information to enter into the publish wizard is the database connection settings for the intrinsic data source. This time, LightSwitch will automatically create the database and publish the schema to SQL Azure. You can start by getting the connection string of the master database that's created automatically for you on SQL Azure. Paste this value into the administrator connection string in the publish wizard. Bring up the connection properties in the publish wizard by clicking on the builder button and enter a new database name to replace "master" (for example, "VisionClinic")—see **Figure 15**. Update the user password (the connection string that was copied contains only a dummy password). Also consider setting the Encrypt connection property to True and the TrustServerCertificate connection property to False in the Advanced Properties for the connection string. This ensures the connection is encrypted and man-in-the-middle attacks aren't possible. You don't want the application to use the administrative connection string for its CRUD operations, so create a new login for the database server by clicking the Create Database Login button on the publish wizard.

Next, you should enter a user name and password for the application's initial security administrator. A security administrator provides access to the security administration screens in the running LightSwitch client application. These screens allow the security administrators to provide initial access to other users to run the application.

You can continue to use the PrescriptionContoso connection string that was previously added on the Other Connections wizard page. On the Specify a Certificate wizard page, you can choose to sign the client application (see **Figure 16**). Signing the application allows the Silverlight out-of-browser launcher to automatically update the application on a user's machine when you publish a newer version of the application to Windows Azure (bit.ly/iY6lFP).

When the wizard is done, the application is packaged up and sent off to Windows Azure for installation. When publishing is complete, LightSwitch will launch the Windows Azure Platform Management Portal. LightSwitch tells Windows Azure to start the hosted service, but it may take a few minutes for the process to complete. You can track the progress of the deployment in the Hosted Service tab of the navigation pane on the portal.

When the deployment has been completed, select Hosted Services in the navigation pane of the portal, and then open child nodes in the items list to reach the completed "Vision_Clinic" deployment (see **Figure 17**). The Properties window should contain a DNS name for your deployed application; clicking on this link will open the Web page to your application, after first automatically redirecting the application to use HTTPS. This is the URL clients should use to run the application. It's possible—but

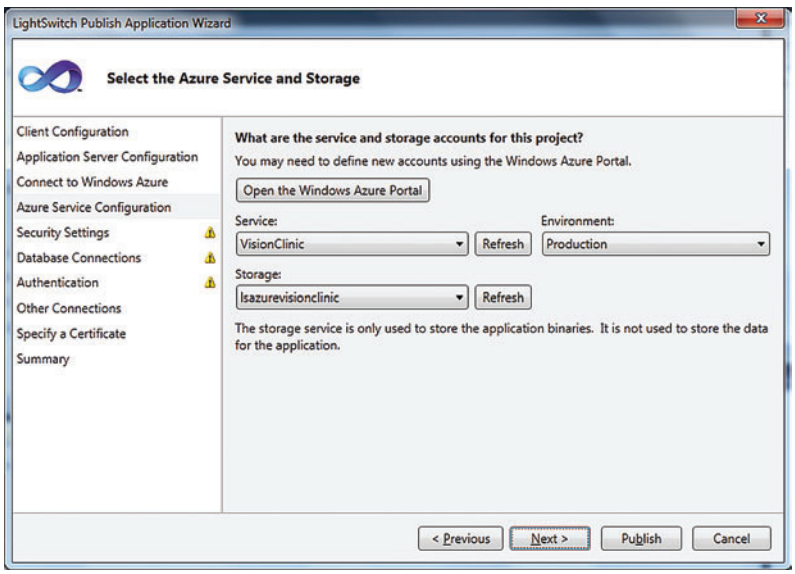


Figure 13 **Setting Windows Azure Service and Storage Information While Publishing**

Our Name Says It All

activePDF®

Come and see why thousands of customers have trusted us over the last decade for all their server based PDF development needs.

. Convert over 400 files types to PDF
. High fidelity translation from PDF to Office
. ISO 32000 Support including PDF/X & PDF/A
. HTML5 to PDF
. True PDF Print Server
. Form Fill PDF
. Append, Stamp, Secure and Digitally Sign

activePDF®
Leading the iPaper Revolution

Compatible with Windows 7 | Windows Vista® | Microsoft Partner Silver Midmarket Solution Provider

Download our FREE evaluation from www.activepdf.com/MSDN

Call 1-866-GoTo-PDF | 949-582-9002 | Sales@activepdf.com

highly discouraged—to turn HTTPS redirection off by setting the Microsoft.LightSwitch.RequireEncryption property to "false" in the service configuration file (described in the following section).

Publish Output

Let's take a look behind the scenes. There are three output files unique to publishing to Windows Azure. They are:

- Vision Clinic.cspkg
- ServiceDefinition.csdef
- ServiceConfiguration.cscfg

In the build output directory, you'll see a .cspkg file. This file is a bundle of all of the files relevant for the Windows Azure application: the application binaries as well as the service definition and configuration files. During the publish process this file is transferred to Windows Azure to configure the hosted service.

ServiceDefinition.csdef (see **Figure 18**) is the Cloud Services definition file for the application. This XML file dictates how the Web site for your application can be configured. The file declares that the one WebRole, LightSwitchWebRole (under the WebRole node), should enable both HTTP and HTTPS (the Sites and EndPoints nodes). It specifies that HTTPS will require a certificate stored on the Windows Azure machine (Certificates node), and declares customizable configuration settings specific to this application (ConfigurationSettings node). These settings include diagnostics logging information as well as whether to redirect HTTP calls to HTTPS.

Figure 19 shows the Cloud Services configuration file (ServiceConfiguration.cscfg) contains the actual configuration settings for the Web Role. If you look at the ServiceConfiguration.cscfg file in the build output directory, you'll see a setting that contains the SSL certificate thumbprint that was specified in the "Security Settings" page of the publish wizard.

There are two ways to change the additional configuration settings for the Web Role. One option is to edit the ServiceConfiguration.cscfg file contained in the LightSwitch project directory. For example, to enable diagnostic logging on the server, you'll need to modify the Microsoft.LightSwitch.Trace.Enabled property, like this:

```
<Setting name="Microsoft.LightSwitch.Trace.Enabled" value="true" />
```

The values set in the project's cloud service configuration file in the project will end up in the built cloud service configuration file.

It's also possible to change the values in ServiceConfiguration.cscfg post-deployment. In the Windows Azure Platform Management Portal, click Hosted Services, Storage Accounts & CDN, then click Hosted Services in the lower part of the navigation pane. Drill down to the Vision_Clinic deployment item in the items list. Once there, you should see the Configure button on the ribbon. Clicking this button allows you to either upload a new service configuration file or edit the existing file.

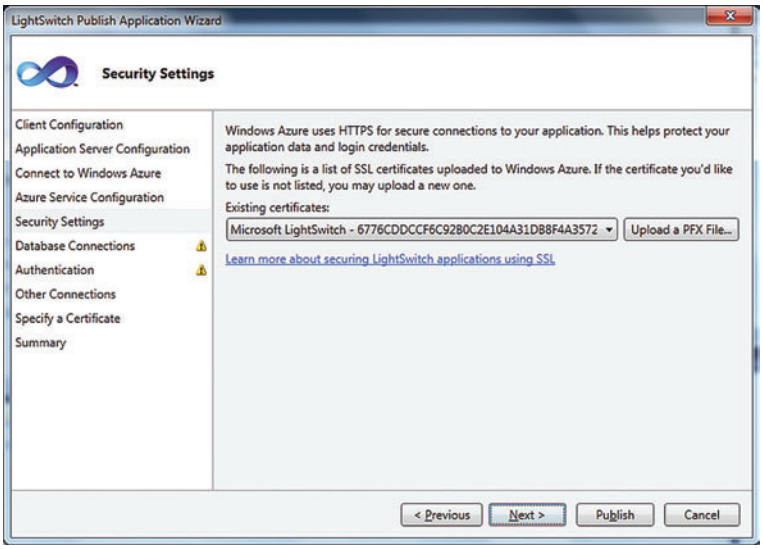


Figure 14 **Setting the Certificate to Use for an HTTPS Connection**

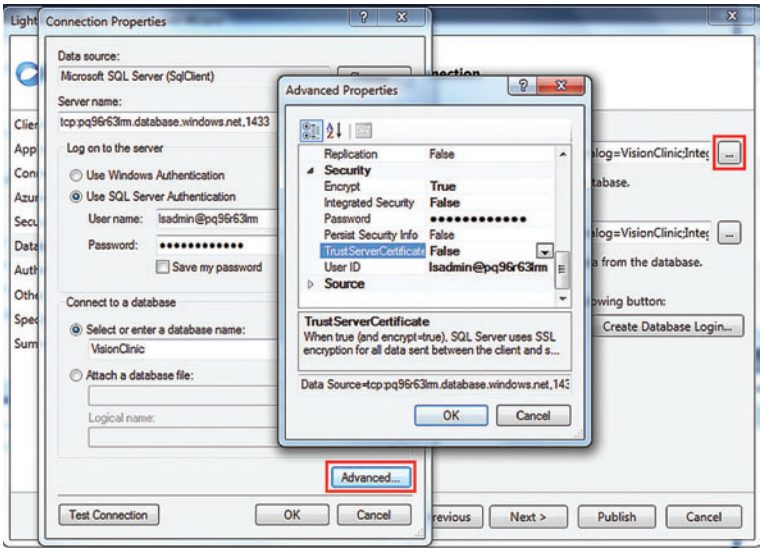


Figure 15 **Setting the Connection String for the Application Data**

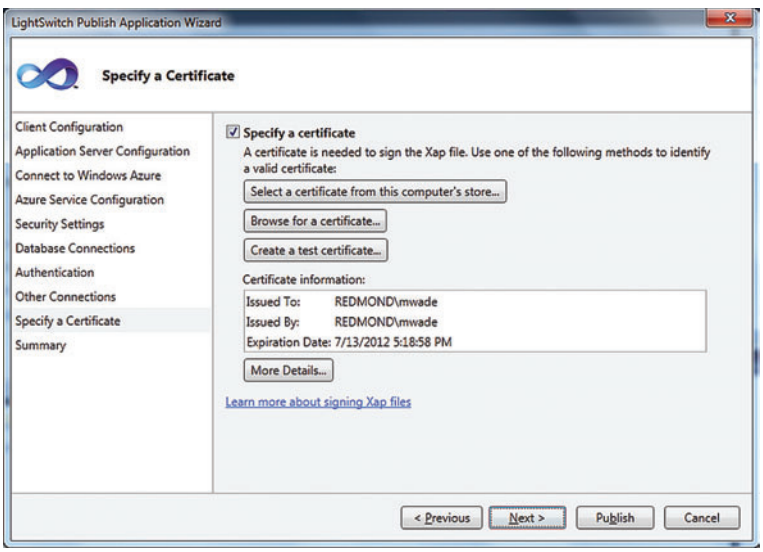


Figure 16 **Specifying a Certificate to Sign the Client Application**

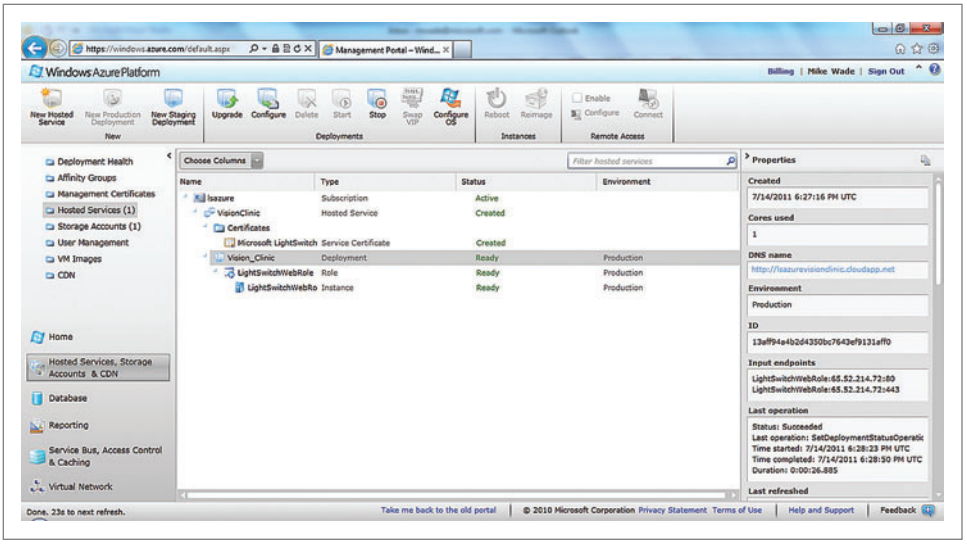


Figure 17 **Viewing the Published Application in the Windows Azure Platform Management Portal**

```
<WebRole name="LightSwitchWebRole"
         vmsize="ExtraSmall"
         enableNativeCodeExecution="true">
```

ExtraSmall (which is currently available in beta release) is the least resource-intensive service available on Windows Azure. A summary of all VM sizes is available at bit.ly/qdTFZp.

Microsoft Windows Azure Connect Windows Azure Connect (bit.ly/nvTg6Z) allows for IP-based network connections between Windows Azure and on-premises resources. Windows Azure Connect, currently in CTP, provides a way for an application deployed to Windows Azure to connect to an on-premises SQL Server database or SharePoint site. You could also have a Windows Azure VM join your local domain, enabling your application to use Windows Authentication. It's possible to configure your LightSwitch application to take advantage of the Windows Azure Connect features. While performing this configuration is beyond the scope of this article, keep an eye on the "Deployment" page of the LightSwitch Developer Center (bit.ly/pxmV5d). ■

Using Additional Windows Azure Features

Although there's no built-in support for many additional Windows Azure features, with some simple modifications to your project's Service Definition.csdef and ServiceConfiguration.cscfg files, it's possible to take advantage of additional Windows Azure features. Here are a couple of examples.

Virtual Machine (VM) Size Windows Azure provides several sizes of compute instances, which dictate the amount of resources dedicated to each service instance: the greater the VM size, the more expensive the application is to run. LightSwitch applications default to a Small compute size, but you can change the level by updating the "vmsize" attribute of the default WebRole in the ServiceDefinition.csdef file:

Figure 18 **The ServiceDefinition.csdef File**

```
<ServiceDefinition name="Vision_Clinic"
  xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceDefinition">
  <WebRole name="LightSwitchWebRole"
           vmsize="Small"
           enableNativeCodeExecution="true">
    <ConfigurationSettings>
      <Setting name="Microsoft.LightSwitch.Trace.Enabled" />
      <Setting name="Microsoft.LightSwitch.Trace.LocalOnly" />
      <Setting name="Microsoft.LightSwitch.Trace.Level" />
      <Setting name="Microsoft.LightSwitch.Trace.Sensitive" />
      <Setting name="Microsoft.LightSwitch.Trace.Categories" />
      <Setting name="Microsoft.LightSwitch.RequireEncryption" />
    </ConfigurationSettings>
    <Sites>
      <Site name="Web">
        <Bindings>
          <Binding name="HttpIn" endpointName="HttpIn" />
          <Binding name="HttpsIn" endpointName="HttpsIn" />
        </Bindings>
      </Site>
    </Sites>
    <Endpoints>
      <InputEndpoint name="HttpIn" protocol="http" port="80" />
      <InputEndpoint name="HttpsIn" protocol="https" port="443"
        certificate="SSLCertificate" />
    </Endpoints>
    <Certificates>
      <Certificate name="SSLCertificate" storeLocation="LocalMachine"
        storeName="My" />
    </Certificates>
  </WebRole>
</ServiceDefinition>
```

Figure 19 **The ServiceConfiguration.cscfg File**

```
<ServiceConfiguration serviceName="Vision_Clinic"
  xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceConfiguration">
  <Role name="LightSwitchWebRole">
    <Instances count="1" />
    <ConfigurationSettings>
      <!-- A value of true will enable diagnostic logging on the server -->
      <Setting name="Microsoft.LightSwitch.Trace.Enabled" value="false" />
      <!-- A value of true only lets local access to Trace.axd -->
      <Setting name="Microsoft.LightSwitch.Trace.LocalOnly" value="true" />
      <!-- The valid values for the trace level are: None, Error,
        Warning, Information, Verbose -->
      <Setting name="Microsoft.LightSwitch.Trace.Level"
        value="Information" />
      <!-- A value of true will indicate that logging sensitive
        information is okay -->
      <Setting name="Microsoft.LightSwitch.Trace.Sensitive" value="false" />
      <!-- The semi-colon separated list of categories that will be
        enabled at the specifed trace level -->
      <Setting name="Microsoft.LightSwitch.Trace.Categories"
        value="Microsoft.LightSwitch" />
      <!-- A value of true will indicate http requests should be
        re-directed to https -->
      <Setting name="Microsoft.LightSwitch.RequireEncryption"
        value="true" />
    </ConfigurationSettings>
    <Certificates>
      <Certificate name="SSLCertificate"
        thumbprint="CD27FF4C85A2AD495A054D606E354BCAAD01B3D8"
        thumbprintAlgorithm="sha1" />
    </Certificates>
  </Role>
</ServiceConfiguration>
```

MIKE WADE *is a developer working on Visual Studio LightSwitch. His focus on the team is deployment and project tooling features.*

THANKS to the following technical experts for reviewing this article: *Beth Massi and John Rivard*

Better Web Forms with HTML5 Forms

Brandon Satrom

If you're a Web developer, you've probably created an HTML Form before. In fact, you may have created more of them than you care to remember. You're no doubt familiar with classic input types like text, password, file, hidden, checkbox, radio, submit and button, and you've probably used most or all of these at various times.

If I were to ask you what type of input—from the previous list—you use more than any other, you'd probably say "text," as would most of us. The text input type is the multitool of classic HTML Forms development. On the one hand, it's able to adapt to nearly any job you give it, but on the other, it's semantically neutral, so the browser offers no help in turning this element into something practical. To compensate, developers and designers have added their own semantics to these elements (via IDs and class names) and have relied on server frameworks and JavaScript to handle validation and add rich interactions.

A classic example is collecting dates in text fields. Most of the time, you want to enhance a date field with a date picker of some kind. This is often done with hand-rolled JavaScript or a framework like jQuery UI, which adds an interaction behavior that allows the user to select a date from a widget and have that date populated into the original field.

As useful as that pattern is—and we've become quite adept as developers with patterns like this—it's repeated so often that you can't help but ask, "Why can't the browser just do it?"

The good news is that, with HTML5 Forms, the browser can—and will. In addition to text and the handful of existing types we've had for years, HTML5 adds 13 new values for the input type attribute, as well as a host of other attributes that will speed up your forms development. This month, I'll share some of the new input types and attributes coming with HTML5 Forms, as well as their implementation status in various browsers. Next, I'll present a quick overview of new client validation features for HTML5 Forms. Finally, I'll take a look at how recent updates to Visual Studio 2010 and the Microsoft .NET Framework 4 enable HTML5 Forms and ASP.NET Web Forms to play well together. Throughout, I'll discuss how you can embrace HTML5 Forms in your applications

This article is discusses a prerelease version of Internet Explorer 10 Platform Preview 2. All information is subject to change.

This article discusses:

- New input types in HTML5
- New content attributes for input types in HTML5
- Form validation in HTML5
- Working with HTML5 Forms and ASP.NET Web Forms

Technologies discussed:

HTML5, Visual Studio 2010, Internet Explorer, ASP.NET MVC, WebMatrix, Modernizr, jQuery UI

Code download available at:

code.msdn.microsoft.com/mag201111HTML5

Figure 1 **A Sample Order Form**

today, while providing fallback solutions for older browsers. All of the demos for this article—which are available online—were built using WebMatrix, a free, lightweight Web development tool from Microsoft. You can try WebMatrix out for yourself at aka.ms/webm.

New Input Types in HTML5

What we know today as HTML5 Forms or HTML5 Web Forms started as Web Forms 2.0, a pre-HTML5 specification authored by a group known as the Web Hypertext Applications Technology Working Group, or WHATWG. Much of the initial work by WHATWG became the starting point for what we now call HTML5, and the Web Forms 2.0 effort is now part of the official HTML5 specification, which you can read at bit.ly/njrWvZ. A significant portion of the specification is dedicated to new types and content attributes for the input element, which you'll find at bit.ly/pEdWFs.

As I mentioned, the specification introduces 13 new input types for use in forms: search, tel, url, email, datetime, date, month, week, time, datetime-local, number, range, color.

Using these new types is simple. Say I want to put a new e-mail field on an order form. As you can see in **Figure 1**, I've modified the WebMatrix Bakery Template Web site's Order page with some additional fields, including e-mail.

For this form, the e-mail field is marked up like so:

```
<input type="email" id="orderEmail" />
```

Notice that the type attribute is equal to "email" instead of "text." The best part about the new HTML input types is that you can use them today and they work, at some level, in every single browser. When a browser encounters one of these new types, one of two things will happen.

If the browser doesn't support the new input types, the type declaration won't be recognized. In this case, the browser will gracefully degrade and treat the element as type="text." You can try this by putting this element on a form and typing "document.getElementById('orderEmail').type" in the Internet Explorer 9 F12 Tools Console. So, you can use these new types today, and if the browser doesn't support a given field, it will continue to work just like a regular text field.

If the browser does recognize a type, however, you'll gain some immediate benefits by using it. For recognized types, the browser adds some type-specific built-in behavior. In the case of the email type, Internet Explorer 10 Platform Preview 2 (PP2) and later will automatically validate any input, and present the user with an error message if the provided value isn't a valid e-mail address, as shown in **Figure 2**.

Figure 2 **Automatic Browser Validation of the Email Input Type**

It's easy to infer the purpose and behavior of each element by its type, so we readily gain another level of semantic richness in our forms. Moreover, some of these new input types let the browser provide richer interactions to users out of the box. For example, if I place the following element on a form and then open that form in a browser that fully supports the date input type, the default interaction is much richer than a plain old text box:

```
<input type="date" id="deliveryDate" />
```

Figure 3 shows what the date type can provide in Opera 11.5. The best part is that all I had to do to get this interaction was specify type="date" and the browser took care of all the manual work I previously had to do in JavaScript to offer this level of functionality.

> The HTML5 specification doesn't dictate how browsers should present these new input types.

It's important to note that the HTML5 specification doesn't dictate how browsers should present these new input types, or how they should present validation errors, so you can expect to see subtle to major differences among browsers. For example, Chrome 13 presents a spinner control for the date rather than a date picker, as shown in **Figure 4** (which, of course, may have changed by the time you read this). You should also know that there's ongoing discussion in the World Wide Web Consortium, or W3C, around browser styling and localization of elements like datetime, date and color. The browsers don't all agree, at present, on how to implement these types, and there's no current customization mechanism within existing implementations that's similar to what you'd find with jQuery UI. Should you choose to implement or experiment with these new types, always be sure to provide a thorough fallback solution. In addition, if consistency of presentation and behavior is important to your users, you may need to apply custom styles, override default behaviors on these controls or use a script-based solution.

Earlier I stated that these fields will still behave as regular text fields, which

is a nice piece of graceful degradation the browsers provide for us. But a date field implemented as a plain text box is clunky, and widely considered a poor user experience. With a little help from Modernizr and jQuery UI, however, you can provide a solution that mixes the best of HTML5 Forms with a nice fallback solution.

You'll recall from my last article (msdn.microsoft.com/magazine/hh394148) that Modernizr (modernizr.com) is a JavaScript library that can help you detect support for HTML5 features in the browser. For this example, I want to use Modernizr to help detect support for the HTML5 date input type and, if it's not supported, use the jQuery UI (jqueryui.com) datepicker widget to provide a similar user experience. Once I've downloaded and added references for Modernizr, jQuery and jQuery UI, I can add fallback support for my date elements with just a few lines of code:

```
if (!Modernizr.inputtypes.date) {
  $("input[type=date]").datepicker();
}
```

The result, as seen in Internet Explorer 10 PP2, is depicted in **Figure 5**.

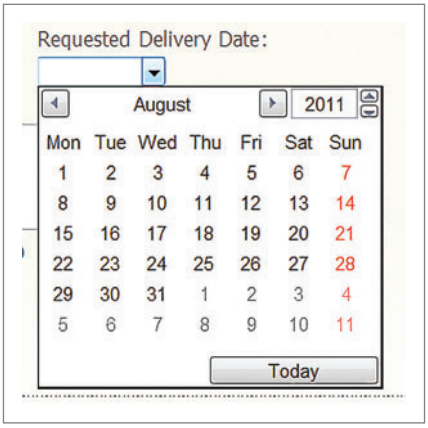


Figure 3 **The Date Input Type in Opera 11.5**

Figure 4 **The Date Input Type in Chrome 13**

New Input Content Attributes in HTML5

In addition to the new input types, HTML5 provides some handy new content attributes that can be used on input fields to supply validation support and enhanced functionality. One of those new attributes is "placeholder," which, according to the specification, "…represents a *short* hint (a word or short phrase) intended to aid the user with data entry" (emphasis theirs). For example, I can take a couple of the fields from our order form and add placeholder="*some text*" to each field, with the result shown in **Figure 6**:

```
<input type="email" id="orderEmail" placeholder="ex. name@domain.com" />
<input type="url" id="orderWebsite" placeholder="ex. http://www.domain.com" />
```

The placeholder text is lighter in color than normal text, and if I place focus on each field, the text disappears, enabling me to enter my own value.

As with the new input types, the placeholder attribute isn't supported in older browsers. Nothing bad will happen if a user visits a page with them, though, so consider using them today, even if you don't plan to add support for older browsers. If you do want to "polyfill" placeholder support, Modernizr can help. As I mentioned in my last article, the good folks at Modernizr try to keep a running list of every polyfill and fallback you could possibly want for a given HTML5 feature. You can check that list out at bit.ly/nZW85d.

For this example, let's use the jQuery HTML5 Placeholder created by Mike Taylor, which you can download from bit.ly/pp9V4s. Once you have it, add the following to a script block referenced by your page:

```
Modernizr.load({
  test: Modernizr.input.placeholder,
  nope: "../js/html5placeholder.jquery.min.js",
  callback: function() {
    $('input[placeholder]').placeholder();
  }
});
```

Here, Modernizr tests whether the placeholder attribute is supported and if it's not, loads html5placeholder.jquery.min.js. jQuery then selects every element with a placeholder attribute and adds plug-in support to each. If you try this out in Internet Explorer 9, you'll notice that the end result looks very similar to the native browser support provided in Internet Explorer 10 PP2.

Another interesting new attribute is "autofocus," which, as it sounds, lets you specify a single form field to automatically receive focus when a page loads. Only one field per page should hold this attribute; if multiple elements are marked up with autofocus, the first one with that declaration receives focus on page load. For my order form, I want the Name field to receive focus, so I add the attribute like so:

```
<input type="text" class="field" id="orderName" autofocus />
```

The autofocus attribute can be used on any form control, and is an excellent alternative to the script-based, form-focused strategies many Web developers have fought with in the past.

HTML5 Form Validation

I don't have the space to cover all the interesting new form-related attributes here, but I'll spend a few moments talking about "required," "pattern," "novalidate" and "formnovalidate," all of which make client-side form validation a snap.

For browsers that support it, the "required" attribute tells the browser that this form can't be submitted without a value. For example, I add "required" to the Name field of my order form:

```
<input type="text" class="field" id="orderName" required />
```

When I visit this page in the Internet Explorer 10 PP2 and attempt to submit the form, I see something like what's shown in **Figure 7**. With a single attribute, the browser knows enough to style the element with a red border and display a message to the user indicating that the field is required.

Earlier, **Figure 2** showed how the browser can automatically validate certain types, such as "email" and "url," without additional input from the user. With the "pattern" attribute, you can provide your own validation test for input types. According to the HTML5 specification, "pattern" expects a regular expression, which the browser uses to validate the owning field.

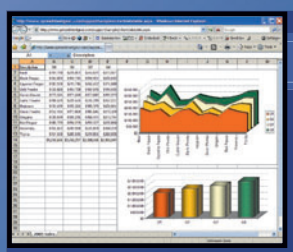
Figure 5 **Date Field Support with jQuery UI**

In a League by Itself

"SpreadsheetGear 2010 is Fantastic! These new capabilities just propelled this control way-way past any competition, of which you have none IMHO. SpreadsheetGear is in a league all by itself."

Greg Newman, Senior Software Engineer, WSFS Cash Connect

SpreadsheetGear 2010

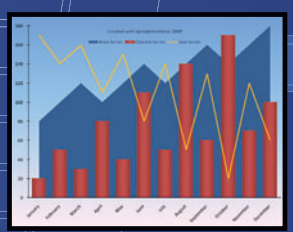


ASP.NET Excel Reporting

Easily create richly formatted Excel reports without Excel using the new generation of spreadsheet technology built from the ground up for scalability and reliability.

Excel Compatible Windows Forms Control

Add powerful Excel compatible viewing, editing, formatting, calculating, charting and printing to your Windows Forms applications with the easy to use WorkbookView control.



Create Dashboards from Excel Charts and Ranges

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

Download the FREE fully functional 30-Day evaluation of SpreadsheetGear 2010 today at

www.SpreadsheetGear.com.

SpreadsheetGear

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

My order form contains a telephone (type="tel") field and I can specify a validation pattern like this:

```
<input type="tel" id="orderTelephone" pattern="\
(\d\d\d\) \d\d\d\-\d\d\d\d" title="(xxx) xxx-xxxx" />
```

This (not very complex) regular expression tells the browser to expect a seven-digit number with parentheses around the area code and a dash in the local number. Entering anything else results in the message shown in **Figure 8**. Notice that the message contains instructions to the user on how to format input: "(xxx) xxx-xxxx." This part of the message is taken from the title attribute of the same element, so it's possible to control at least part of the validation message via your markup. There's one thing to note when using title to aid in validation, though. According to the spec, the browser may choose to show the title in other, non-error cases, so don't count on this attribute as a place for error-sounding text.

Figure 6 **Using the Placeholder Attribute with Input Fields**

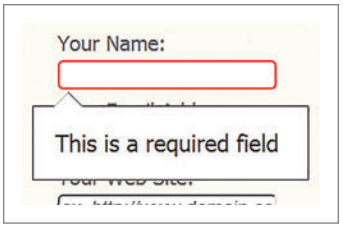


Figure 7 **Using the Required Attribute on a Form Field**

Automating validation by the browser is nice, but two immediate questions come to mind. First, what about server validation or client validation scripts generated by my server framework (ASP.NET MVC, for example)? And second, what about cases where I want the user to be able to save the form as a work in progress, without validation? The first is, unfortunately, outside the scope of this article, but I've written a blog post about this very subject in ASP.NET MVC, which you can find at bit.ly/HTML5FormsAndMVC.

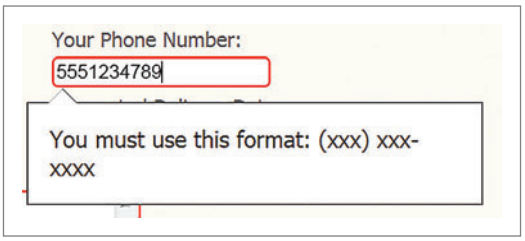


Figure 8 **Using the Pattern Attribute to Specify a Validation Expression**

> The addition of Web Forms support to browsers with HTML5 is good news for Web developers everywhere.

The second question, on the other hand, is easy. Let's assume you have a form that users will spend quite a bit of time on before submitting, perhaps even saving multiple times before finally posting it to your server. For such cases, where you'll want to allow a user to submit a form without validation, there are two attributes you can use: "formnovalidate," which is placed on input fields of type "submit," and "novalidate," which is placed on an opening form tag. Here I'll place two submit fields on my form, like so:

```
<input type="submit" value="Place Order" />
<input type="submit" formnovalidate value="Save for Later" id="saveForLater" />
```

The "formnovalidate" attribute on the second button will turn off validation and submit the form, allowing the user's work in progress to be saved in my database, or even on the client side using a new HTML5 storage technology like localStorage or IndexedDB.

HTML5 Forms and ASP.NET Web Forms

Before I wrap up this article, I want to share a few additional bits of information related to HTML5 Forms for ASP.NET Web Forms developers. If you're planning to do HTML5 Forms development with ASP.NET Web Forms, there's good news: Many HTML5-related updates to .NET and Visual Studio are being released out-of-band, so you don't have to wait for the next framework version to use these features today.

To get started with HTML5 Forms and ASP.NET Web Forms, you'll need to grab a couple of updates. First, make sure you have Visual Studio 2010 SP1 (bit.ly/nQzsId). In addition to adding support for new HTML5 input types and attributes, the service pack also provides some updates that enable you to use the new HTML5 input types on the TextBox server control. Without this update, you'd see compile-time errors when using the new types.

You'll also want to grab the Microsoft .NET Framework 4 Reliability Update 1 (bit.ly/qOG7Ni). This update is designed to fix a handful of problems related to using the new HTML5 input types with ASP.NET Web Forms. Scott Hunter covers a few of those—UpdatePanel, Validation Controls and Callbacks—in a blog post from early August that you can check out at bit.ly/qE7jLz.

The addition of Web Forms support to browsers with HTML5 is good news for Web developers everywhere. Not only do we have a set of semantic input types to leverage in our applications, but we can also use these input types today with no ill effects in older browsers, while getting enhanced functionality—including automatic client validation—in newer ones. Using these new fields right away has immediate benefits in the mobile application space as well, where using types like "url" and "email" will prompt mobile devices to present the user with soft keyboard options tuned for that input type. When you combine these new features with Modernizr and one of the great polyfilling options, you have all the tools you need to adopt HTML5 Forms in your applications right now.

For more information on HTML5 Forms support in Internet Explorer 10 PP2, go to ietestdrive.com, and be sure to check out the developer's guide at the Internet Explorer Developer Center (bit.ly/r5xKhN). For a deeper dive into HTML5 Forms in general, I recommend "A Form of Madness," from Mark Pilgrim's book "HTML5: Up and Running" (O'Reilly Media, 2010), as well as the Forms section of the W3C HTML5 specification (bit.ly/nIKxfE). ■

BRANDON SATROM *works as a developer evangelist for Microsoft outside of Austin, Texas. He blogs at userinexperience.com and can be found on Twitter as @BrandonSatrom.*

THANKS to the following technical experts for reviewing this article: *Jon Box, John Hrvatin, Scott Hunter and Clark Sell*

Manage Project Libraries with NuGet

Phil Haack

Try as it might, Microsoft can't supply every possible library developers need. Though Microsoft employs close to 90,000 people worldwide, the world has millions of developers. It doesn't make sense for Microsoft to attempt to fill every niche, nor does it scale. Developers, therefore, frequently take it upon themselves to "scratch their own itch," and they've written thousands upon thousands of useful libraries that are distributed all over the Web.

The problem with so many libraries out there is sharing them. Sharing and reusing code is a big challenge. Don't believe me? Walk into any midsize to large shop and ask them how many logging libraries they have. Visit enough companies and you'll find a high percentage of in-house logging libraries when good ones—Log4Net, NLog and Error Logging Modules and Handlers, or ELMAH—already exist.

When a developer starts a new project, he faces the empty canvas problem. How can he find these useful libraries? How does he incorporate a library into his current project and manage its dependencies and updates?

ELMAH is a great example of a useful library that developers took upon themselves to write. ELMAH logs all unhandled exceptions within a Web application along with all the request information, such as headers, server variables and so on, when the exception is thrown. Suppose you've just heard about ELMAH and want to use it in your next project.

These are the steps you might take:

1. **Find ELMAH.** Due to its unique name, a Bing search locates the ELMAH Google code page as the first result.
2. **Download the correct zip package.** The download page for the site has multiple zip packages. You have to think about it and pick the correct one. Sometimes the correct choice isn't intuitive.
3. **"Unblock" the package.** Once you've downloaded the package from the Web, you need to right-click the file, bring up the Properties dialog and click the Unblock button to remove the "mark of the Web" from the file.
4. **Verify its hash against the one provided by the hosting environment.** The Google code site displays a QR code representing the zip file. How many developers do you know who take the time to verify the file against the QR code?
5. **Unzip the package contents into a specific location in the solution.** Most developers avoid unpacking assemblies into the bin directory, because the directory is

This article discusses:

- The challenge of sharing and reusing code
- Adding and updating libraries in a Visual Studio project with NuGet
- Using Windows PowerShell with NuGet
- Extending Visual Studio with new commands
- Using NuGet within the enterprise

Technologies discussed:

NuGet, Visual Studio, Windows PowerShell, ASP.NET

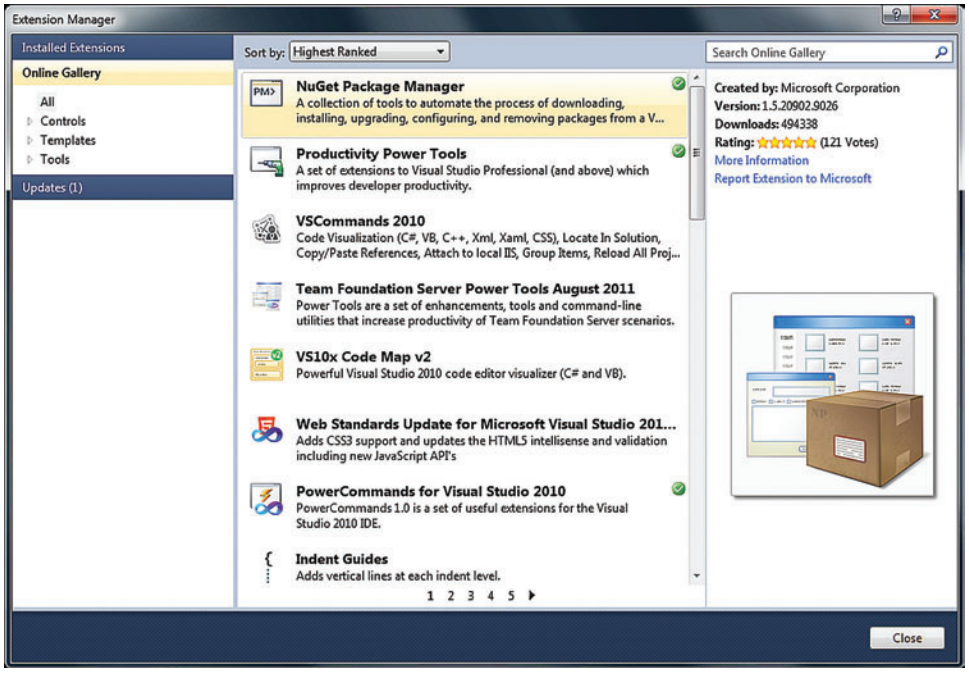


Figure 1 **Visual Studio Extension Manager**

meant for output from the build, not for input, and isn't tracked by version control. Instead, it's important to add the dependency to a folder that's committed to version control and reference the assembly from that location.

6. **Add an assembly reference to the project.** The assembly isn't usable until you add a reference to it from within the Visual Studio project.
7. **Update web.config with the correct settings.** This may mean more searching around using Bing or Google while you try to find the correct settings needed for your config file.

What a pain! Now suppose you have to do this for 10 to 15 dependencies. When it comes time to release the next version of your application, you spend significant time searching for updates to your application's dependencies.

That often-maligned notion, "not invented here" (NIH), begins to sound like a good idea.

NuGet to the Rescue

NuGet is a Visual Studio extension that makes it easy to add, update and remove libraries (deployed as packages) in a Visual Studio project. A NuGet package is a set of files packaged up into a single file with the .nupkg extension using the Open Packaging Conventions (OPC) format.

OPC is just a fancy acronym for a zip file with some metadata. In fact, you're probably already familiar with OPC, as it's the format used by Word and Excel documents. If you've ever taken a .docx file and changed the file extension to .zip, you know you can open it up and poke around its internals. The same goes for .nupkg files.

The NuGet product also comes with utilities to easily create and publish packages. For now, I'll focus on using NuGet to discover and install packages. Later I'll cover how to create and publish packages.

Installing NuGet

To install NuGet, launch the Visual Studio Extension Manager via the Tools | Extension Manager menu option. Click the Online Gallery tab to view available Visual Studio extensions, as shown in **Figure 1**. As you can see, NuGet is the highest-ranked package, which puts it on the first screen. If that ever changes, you can use the search box on the top right to find it. Click the Download button to install NuGet.

If you've installed ASP.NET MVC 3, you already have NuGet installed. ASP.NET MVC 3 includes NuGet, and Microsoft plans to include NuGet in the next version of Visual Studio.

Installing a Package

Let's start with NuGet's user-friendly dialog to install packages. NuGet also comes with a Windows PowerShell-based console geared toward power users that I'll cover later.

To launch NuGet, right-click on the project's references node and select the Manage NuGet Packages option (this option had a different label prior to NuGet 1.4). This launches the Manage NuGet Packages dialog shown in **Figure 2**.

Make sure the Online tab is selected and type in a search term in the top right (for example, search for MiniProfiler, a useful library from the StackOverflow.com folks).

Once you locate a package, click the Install button to install the package. NuGet then downloads the package and its dependencies and applies any necessary changes to your project specified by the packages.

NuGet performs the following steps to install a package:

1. **Downloads the package file and all its dependencies.** Some packages require explicit license acceptance and prompt the user to accept the license terms for the package. Most packages are fine with implicit license acceptance and do not prompt. If the package already exists in the solution or in the local machine cache, NuGet skips downloading the package.
2. **Extracts the package's contents.** NuGet extracts the contents into the packages folder, creating the folder if necessary. The packages folder is located next to your solution (.sln) file. If the same package is installed into multiple projects within a solution, the package is only extracted once and is shared by each project.
3. **References assemblies in the package.** By convention, NuGet updates the project to reference the appropriate assembly (or assemblies) within the packages *lib* folder. For example, when installing a package into a project targeting the Microsoft .NET Framework 4, NuGet will add references to assemblies within the *lib/net40* folder.

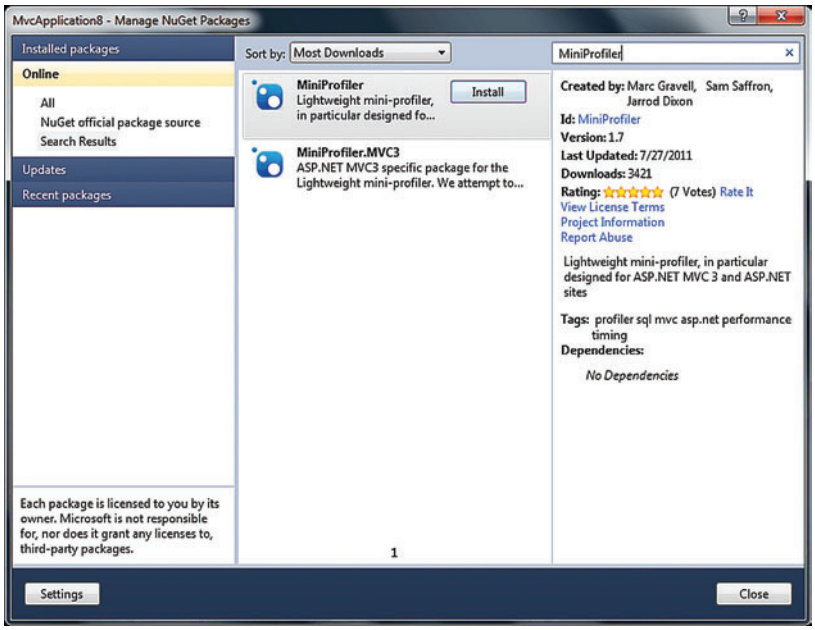


Figure 2 **NuGet Package Manager Dialog**

4. **Copies content into the project.** By convention, NuGet copies the package's content folder's contents into the project. This is useful for packages containing JavaScript files or images.
5. **Applies package transformations.** If any package contains transform files, such as app.config.transform or web.config.transform for config, NuGet applies those transformations before it copies the content. Some packages may contain source code that can be transformed to include the current project's namespace in the source file. NuGet transforms those files as well.
6. **Runs associated Windows PowerShell scripts in the package.** Some packages may contain Windows PowerShell scripts that automate Visual Studio using the Design Time Environment (DTE) to handle tasks NuGet isn't designed for.

After NuGet performs all these steps, the library is ready for use. Many packages wire themselves up using the WebActivator package to minimize any configuration necessary after installation.

A package can be uninstalled, which returns your project to the state it was in before installing the package.

Updating Packages

In his book, "Facts and Fallacies of Software Engineering" (Addison-Wesley Professional, 2002), Robert L. Glass states: "Maintenance typically consumes about 40 to 80 percent (60 percent average) of software costs. Therefore, it is probably the most important lifecycle phase."

Installing a package is only the beginning of the story. Over time, as these libraries are maintained, it becomes important to keep your application up-to-date with the latest bug fixes for the libraries. This requires you to answer the question, "Which dependencies in this project have new updates available?"

As mentioned before, answering this question has always been a time-consuming endeavor. With NuGet, however, you simply launch the dialog and click on the Updates tab to see a list of packages with available updates, as shown in **Figure 3**. Click the Update button to update the package to the latest version.

The update command uninstalls the old package and then installs the new one, ensuring all dependencies are updated appropriately if needed.

NuGet provides commands in the Package Manager Console to better control updates—such as to update all packages in the solution or to perform a "safe" update, for example.

NuGet for Power Users

While I'm a big fan of nice GUI dialogs, I know many developers who disdain mouse-dragging people like me. Those folks prefer a command-line shell as the UI for its ability to compose sets of commands together.

NuGet fills this need with a Windows PowerShell-based console window called the Package Manager Console, as well as a set of Windows PowerShell commands for interacting with NuGet. Windows PowerShell, a .NET-based scripting language and command-line shell, is well-suited for composing command sets and working with objects.

To launch the Package Manager Console, navigate to the Tools | Library Package Manager | Package Manager Console menu option.

Listing and Installing Packages

To list and search for packages, use the Get-Package command. By default, the command lists installed packages in the current project. You can search for packages online by specifying the ListAvailable flag combined with the Filter flag. The following command searches for all packages mentioning "MVC":

```
Get-Package -ListAvailable -Filter Mvc
```

Once you find a package to install, use the Install-Package command. For example, to install ELMAH into the current project:

```
Install-Package Elmah
```

Because Windows PowerShell is a dynamic language, it can provide tab expansions to help you correctly enter command arguments. Tab expansions are equivalent to IntelliSense for C# but, unlike IntelliSense, which is based on compile-time information, tab expansions can be populated at run time.

For example, if you type in Install-Package Mvc{tab}, you'll see a list of possible package names from the package source, as shown in **Figure 4**.

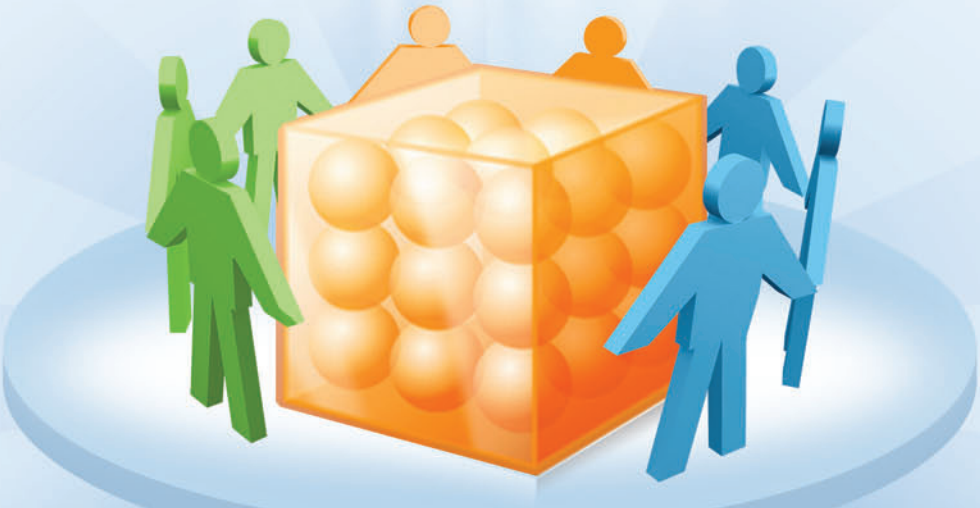
Updating Packages

The Package Manager Console also includes a command that provides more control over updates than the dialog. For example, call this command with no arguments to update every package in every project of the solution:

```
Update-Package
```

InstallAware

NEW INSTALLAWARE 2012



App-V for the Masses

Visit www.installaware.com/landing/msdn.asp for this special pricing

SPECIAL

Instantly Build
App-V Packages

for Only
$890

Windows 8 Ultimate

NOW WITH SUPPORT FOR WINDOWS 8

App-V Builder:
Compiles any existing InstallAware project as an App-V Package

App-V Viewer:
Opens and inspects the contents of any pre-existing App-V Package

Hybrid 32 bit & 64 bit:
Combines both 32 bit and 64 bit applications inside a single App-V

MSI Deployment:
Creates an MSI file to silently push your App-V Packages

Command Line Support:
Automates your App-V Build process through the command line

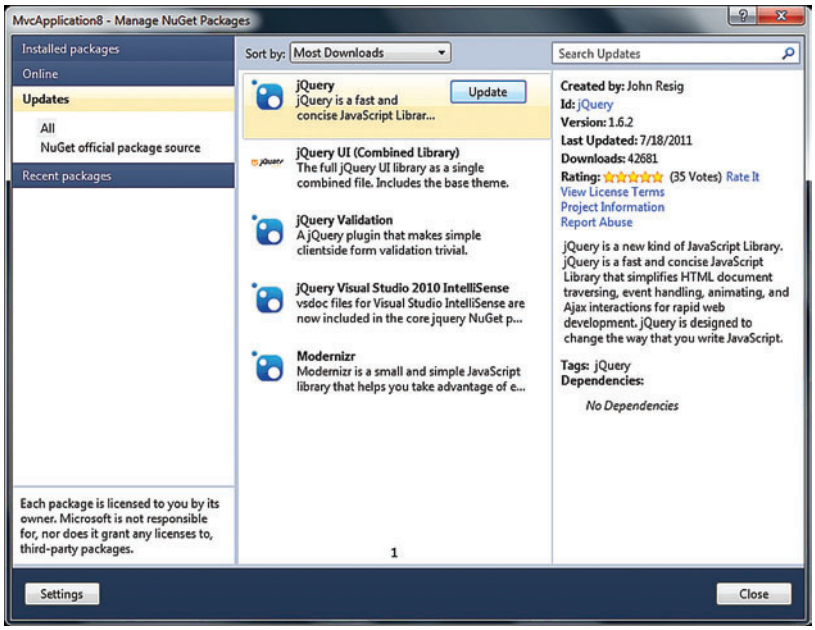


Figure 3 **Updates Available for the Current Project**

This command attempts to update each package to the latest version. Thus, if you have version 1.0 of a package and versions 1.1 and 2.0 are available in the package source, this command will update the package to version 2.0, because it's the latest.

This can be a drastic action if any package contains breaking changes. In many cases, you'll just want to update every package to the latest bug fix release. This is called a "safe" update and assumes that packages with a larger build or revision number (but with the same major and minor numbers) are backward-compatible. Just add the Safe flag to perform a safe update, like so:

```
Update-Package -Safe
```

In this case, if you have version 1.0.0 of a package installed, and both 1.0.1 and 1.1 are available in the package source, the package is safely upgraded to 1.0.1 and not 1.1.

The Update-Package command also provides more granularity, such as updating a package to a specific version of the package rather than the latest version.

Extending Visual Studio with New Commands

While the ability to use Windows PowerShell to install packages is nice, it's not the most compelling reason to choose Windows PowerShell. One of the most compelling reasons is that packages can add new commands to the Package Manager Console. These commands can interact with the Visual Studio DTE in order to perform various tasks.

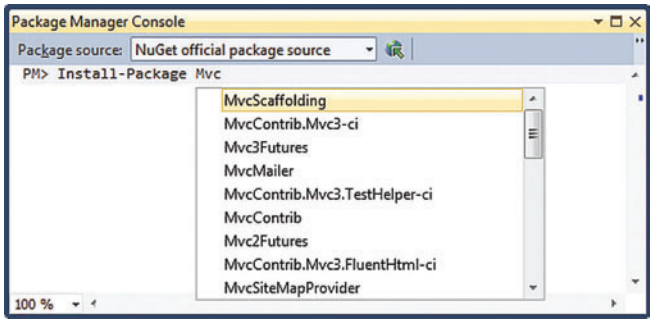


Figure 4 **A Tab-Expanded List of Packages**

For example, install the MvcScaffolding package and it adds a new Scaffold Controller command to the console. Given an Entity Framework (EF) Code First object, this command generates a controller, controller actions and views corresponding to the basic Create, Read, Update and Delete (CRUD) operations for the EF object, as seen in **Figure 5**.

NuGet in Your Organization

With all this focus on how NuGet makes it easy to share libraries with the public developer community, NuGet's usefulness within the enterprise is easy to miss.

After all, there's nothing special about businesses that make them immune to the same challenges the software community as a whole faces when sharing code. As a company grows over time, entropy sets in. Different groups within the same company use their own private versions of company "standard" libraries. Some groups may go so far as to completely ignore these libraries and write their own from scratch.

Often, the problem is not the libraries themselves but the hassle in sharing these libraries with other teams and keeping them notified of changes. Sound familiar?

Package Sources

So far, I've covered how to install packages, but haven't answered the obvious question: Where are those packages located? They're located in the official NuGet package gallery at nuget.org. This gallery exposes an OData feed: packages.nuget.org/v1/FeedService.svc.

The OData format allows the NuGet client to generate ad hoc queries to search the package source on the client, but have them executed on the server.

To add more package sources to NuGet, navigate to the Tools | Library Package Manager | Package Manager Settings menu option and click on the Package Sources node, as seen in **Figure 6**.

The default package source is an OData endpoint on the Web, but the example screenshot also shows a local folder as a package source. NuGet treats folders (whether local or on a network share) as a package source and lists each package within the folder in the Online pane. This makes sharing code with others as easy as putting it in a folder, and is also helpful when you test packages you create yourself.

Hosting Your Own NuGet Server

In addition to hosting packages on a network share, you can also set up a Web site as a package source and use it to share packages with others across your organization.

As with many tasks, there's a package that helps here. First, create an empty ASP.NET Web Application in Visual Studio (targeting ASP.NET 4). Use NuGet to install the package NuGet.Server. This package adds a simple OData endpoint to the empty Web application.

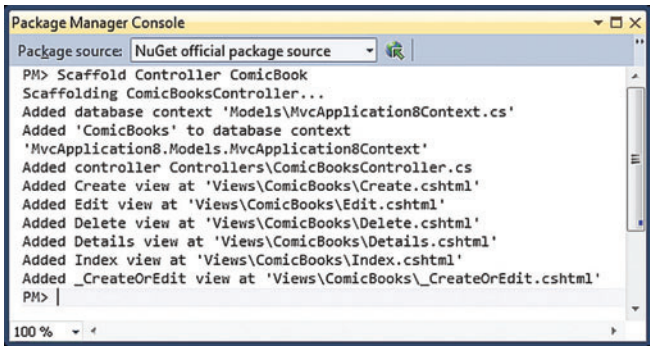


Figure 5 **MvcScaffolding Custom Scaffold Command in Action**

Next, add package files to the Web application's Packages folder to publish them and deploy the Web site. For more details on how to set this up, refer to the NuGet documentation site, bit.ly/jirmL0.

For those who want to deploy a full gallery experience like nuget.org, the NuGet gallery code is also available as an open source project via the nugetgallery.codeplex.com project.

Hosting a private NuGet server or gallery implementation is an easy way to share proprietary code within a company without having to publish it to the public.

Creating a Package

NuGet wouldn't be useful without any packages to install in the first place. The more useful packages available in NuGet, the more valuable NuGet becomes to every developer. That's why the NuGet team has gone to great lengths to make creating packages as simple and painless as possible. While it's easy to create a package, the NuGet team continues to invest in features to make it even simpler. They've built several tools for creating NuGet packages. For example, Package Explorer is a ClickOnce application written by a developer on the NuGet team that provides a visual way to build or examine a package. It's available at npe.codeplex.com.

NuGet.exe

Because most package authors want to integrate package creation into their build processes, let's look at another approach that uses the NuGet command-line utility. You'll need to download the utility just once from bit.ly/gmw54b. After you download NuGet.exe, make sure to put it in a folder that's added to your PATH environment variable. I have a folder named "utils" for utilities like this.

The reason I say you only need to download NuGet.exe once (well, once per machine) is because it's a self-updating executable. Just run the following command to have NuGet check online and update itself to the latest version if a newer one is available:

```
nuget update -self
```

The command-line tool can query the online feed like the Package Manager Console. For example, to search for all packages with "MVC," use the following command:

```
nuget list Mvc
```

NuGet.exe can even download a package and dependencies and unpack them, but it can't modify a project to reference the downloaded package assemblies or run any Windows PowerShell scripts included in a package.

Creating a Package from a Project

Packages contain a single assembly 90 percent of the time (a statistic I made up). This section covers a simple workflow to create such packages using NuGet.exe against a project file (such as a .csproj or .vbproj file).

For details on creating more complex packages, such as a single package that targets different .NET Framework versions, refer to the NuGet documentation Web site at docs.nuget.org.

The basic steps for creating a package are:

1. Create a class library project.
2. Generate a NuSpec manifest from the project.
3. Update the project's assembly metadata.
4. Use NuGet.exe to create the package.

Create a Class Library Project To share an assembly, start with a class library project. NuGet can pack multiple project types, but the most common case when sharing code is to use a class library. After you create the package, make sure to open the AssemblyInfo.cs file to update the assembly's metadata.

Create a NuSpec Manifest The NuSpec file is a package manifest with important metadata about the package, such as the author, description and the package dependencies. The NuSpec format is simple to create by hand, but it's more convenient to let the spec command generate the file for you. Make sure to run the command in the same directory as the project file:

```
nuget spec
```

In this particular case, because the spec command generated the NuSpec from a project file, it contains placeholders for some metadata, as shown in **Figure 7**.

Don't edit the fields with the placeholders, but do fill in the correct values for the other fields such as licenseUrl, projectUrl, iconUrl and tags.

Update the Project's Assembly Metadata Every assembly has metadata related to the assembly. NuGet can read this assembly metadata and merge it into the NuSpec manifest

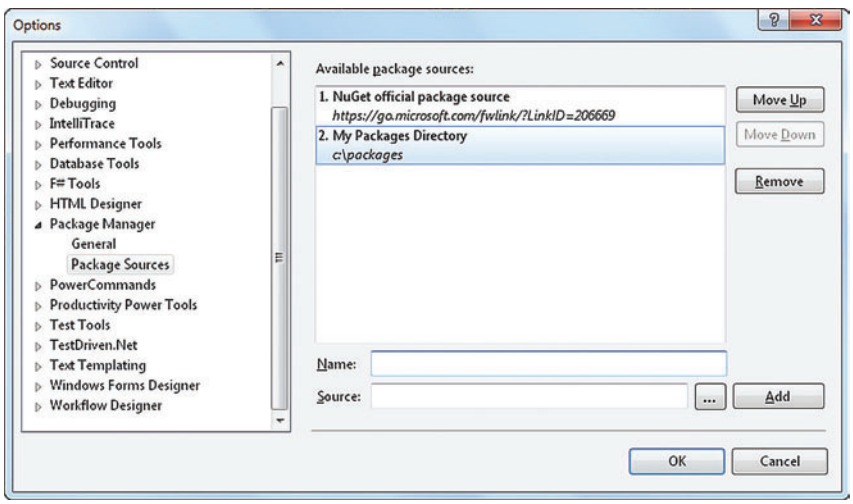


Figure 6 **Package Manager Settings**

Figure 7 **The Generated NuSpec File**

```
<?xml version="1.0"?>
<package xmlns=
  "http://schemas.microsoft.com/packaging/2010/07/nuspec.xsd">
  <metadata>
    <id>$id$</id>
    <version>$version$</version>
    <title>$title$</title>
    <authors>$author$</authors>
    <owners>$author$</owners>
    <licenseUrl>http://LICENSE_URL_HERE_OR_DELETE_THIS_LINE</licenseUrl>
    <projectUrl>http://PROJECT_URL_HERE_OR_DELETE_THIS_LINE</projectUrl>
    <iconUrl>http://ICON_URL_HERE_OR_DELETE_THIS_LINE</iconUrl>
    <requireLicenseAcceptance>false</requireLicenseAcceptance>
    <description>$description$</description>
    <copyright>Copyright 2011</copyright>
    <tags>Tag1 Tag2</tags>
  </metadata>
</package>
```

when it creates a package, which ensures this information is never out of sync between your package and your assembly.

As mentioned earlier, this information is usually located in a file named AssemblyInfo.cs. The table in **Figure 8** shows the mappings between the assembly metadata and the NuSpec placeholder values.

Unlike the other fields, the id field is not extracted from an assembly attribute, but is set to the assembly name.

Create the Package In the same directory as the project file and NuSpec file, run the following command to create a package:

```
nuget pack ProjectName.csproj
```

If you have one project file in the same directory, you can omit the project file name when you run the command.

If you haven't compiled the project yet, you can use the Build flag to compile the project first, before packing it. This will compile the project first before running the pack command:

```
nuget pack ProjectName.csproj -Build
```

This command results in a file named ProjectName.{version}.nupkg, where {version} is the same value specified in the AssemblyVersionAttribute. For example, if the version is 1.0.0, your package will be named ProjectName.1.0.0.nupkg. You can use the Package Explorer to examine the package after the fact to ensure it's been created correctly.

As a courtesy to developers who will install your package, consider using the Symbols flag to create a package with debugger symbols:

```
nuget pack ProjectName.csproj -Build -Symbols
```

This command creates a symbols package in addition to the main package. This allows others who install your package to step into the package code when they debug their application.

Figure 8 **Assembly Metadata Mapped to NuSpec**

Token	Source
id	The assembly name.
$title$	The assembly title as specified in the AssemblyTitleAttribute.
$version$	The assembly version as specified in the assembly's AssemblyVersionAttribute.
$author$	The company as specified in the AssemblyCompanyAttribute.
$description$	The description as specified in the AssemblyDescriptionAttribute.

Publishing a Package

After you create a package, you'll probably want to share it with the world. NuGet.exe has a publish command for just this purpose. Before you publish, you'll need to create an account on nuget.org.

When you've registered for an account, click on the link to your account to see your access key. This key is important as it identifies the nuget.exe command to the gallery and is a revocable password.

Once you have your key, store it in a secure location using the following command:

```
nuget setApiKey b688a925-0956-40a0-8327-ff2251cf5f9a
```

With this in place, use the push command to publish your package to the gallery:

```
nuget push ProjectName.1.0.0.nupkg
```

The command validates your API key with the gallery before it uploads the package. If you created a symbols package as we discussed earlier, you should specify the Symbols flag when you push your package:

```
nuget push ProjectName.1.0.0.nupkg -Symbols
```

Be sure to specify the main package name and not the symbols package name. The command finds the appropriate symbols package by convention. The command pushes the main package to the NuGet gallery and the symbols package to the partner symbolsource.org repository.

What's Next

In this article, I demonstrated how NuGet pulls in useful libraries from the NuGet gallery to jump-start new project development. Within enterprises, NuGet is useful for sharing code among various developers in an organization.

But there's one persistent misperception about NuGet I need to address—that NuGet is only for Web developers. This misperception is probably due to its inclusion with the ASP.NET MVC 3 release, but it's simply not true. NuGet is not just for Web developers—it's for all developers. NuGet supports Windows Phone, Silverlight and Windows Presentation Foundation, among other project types, and will support new project types in the future.

NuGet is a community-driven open source project licensed under the Apache 2 license. The project belongs to the Outercurve Foundation but is incorporated into Microsoft products and counts several Microsoft developers as core contributors.

To help in NuGet's development, visit nuget.codeplex.com to learn about how to get involved and maybe even contribute to NuGet.

This article only scratches the surface of what's possible with NuGet. To learn more, visit the NuGet documentation Web site at docs.nuget.org (the team works hard to maintain this and accepts contributions to its documentation). The team is active in the CodePlex discussions forum for the NuGet project and welcomes your questions. ■

PHIL HAACK *works for Microsoft as a senior program manager on the ASP.NET team aiming to build great products for developers. While he delves into many areas of ASP.NET, his primary projects are ASP.NET MVC and NuGet Package Manager, both released under an OSS license. In his spare time, he writes about software on his blog, haacked.com, and works on the Subtext open source blog engine.*

THANKS to the following technical expert for reviewing this article: *David Fowler*

Visual Studio LIVE!

EXPERT SOLUTIONS FOR .NET DEVELOPERS

DECEMBER 5-9
ORLANDO, FLORIDA
Royal Pacific Resort at Universal Orlando®

5 CODE-FILLED DAYS IN FLORIDA!

INTENSE TRAINING + AN AWESOME LOCATION:

Visual Studio Live! Orlando is a must-attend event for **developers, software architects and designers** that provides the tools, technologies and tips you need to solve development challenges. Educational tracks lead by IT experts and .NET rock stars include:

- **Visual Studio 2010 / .NET**
- **Silverlight / WPF**
- **Web / HTML5**
- **Windows Phone 7**
- **Developing Services**
- **Data Management**
- **Cloud Computing**
- **Programming Practices**

REGISTER TODAY!
VSLIVE.COM/ORLANDO

Use Promo Code **NOVAD**

PLATINUM SPONSORS

Microsoft

Syncfusion
Deliver innovation with ease

SUPPORTED BY

Microsoft Visual Studio

msdn

Visual Studio MAGAZINE

PRODUCED BY

1105 MEDIA

VISUAL STUDIO LIVE! ORLANDO

Visual Studio 2010 / .NET 4	Developing Services	Windows Phone 7	Cloud Computing	Data Management	Programming Practices	Web / HTML5	Silverlight / WPF

Visual Studio Live! Pre-Conference Workshops: Monday, December 5, 2011 *(Separate entry fee required)*

MWK1 **Workshop: SQL Server for Developers** *Andrew Brust & Leonard Lobel*	MWK2 **Workshop: Making Effective Use of Silverlight and WPF** *Billy Hollis & Rockford Lhotka*	MWK3 **Workshop: Programming with WCF in One Day** *Miguel Castro*

Visual Studio Live! Day 1: Tuesday, December 6, 2011

Keynote *Microsoft TBA*

T1 **HTML5 & CSS3 Mini-Bootcamp for ASP.NET Developers** *Todd Anglin*	T2 **Intense Intro to Silverlight** *Billy Hollis*	T3 **AppFabric, Workflow and WCF - The Next Generation Middleware** *Ron Jacobs*	T4 **So Many Choices, So Little Time: Understanding Your .NET 4 Data Access Options** *Lenni Lobel*
T5 **The HTML5 Mullet: Form Input and Validation** *Todd Anglin*	T6 **XAML: Achieving Your Moment of Clarity** *Miguel Castro*	T7 **What's New in WCF 4** *Ido Flatow*	T8 **Microsoft Session** *TBA*

Birds-of-a-Feather Lunch

T9 **Chalk Talk: What's New and Cool in Silverlight 5** *Pete Brown*		T10 **Chalk Talk: Building Applications Using CSLA .NET** *Rockford Lhotka*	
T11 **Learning MVC - for the Web Forms Developer** *Adam Tuliper*	T12 **Fundamental Design Principles for UI Developers** *Billy Hollis*	T13 **Creating Scalable Stateful Services Using WCF and WF** *Marcel de Vries*	T14 **Living the Dream: Make the Video Game You've Always Wanted and Get Paid for It!** *Dave Isbitski*
T15 **MVC, Razor, and jQuery - The New Face of ASP.NET** *Ido Flatow*	T16 **Bind Anything to Anything in XAML** *Rockford Lhotka*	T17 **AppFabric Caching: How It Works and When You Should Use It** *Jon Flanders*	T18 **Microsoft Session** *TBA*

Visual Studio Live! Day 2: Wednesday, December 7, 2011

Keynote: **Cloud Applications Today on the Devices of Tomorrow** *Scott Cate, CTO, EventDay.com*

W1 **Creating a Data Driven Web Site Using WebMatrix and ASP.NET Razor** *Rachel Appel*	W2 **Javascript and CSS 3 Patterns for HTML5** *John Papa*	W3 **If Not IaaS, When Should I Use Windows Azure VM Role?** *Eric D. Boyd*	W4 **Team Foundation Server Build Automation Inside Out** *Marcel de Vries*
W5 **Hack Proofing Your ASP.NET Web Forms and MVC Applications** *Adam Tuliper*	W6 **A Lap Around WPF v.next** *Pete Brown*	W7 **What is Windows Azure Marketplace DataMarket?** *Michael Stiefel*	W8 **Implementing Custom Shells, Silverlight Custom Controls and WCF RIA Services in LightSwitch** *Michael Washington*

Lunch

W9 **Chalk Talk: Advanced Patterns with MVVM in Silverlight and Windows Phone 7** *John Papa*		W10 **Chalk Talk: Building RESTful Services with WCF** *Jon Flanders*	
W11 **How Orchard CMS Works** *Rachel Appel*	W12 **Light Up on Windows 7 with Silverlight and WPF** *Pete Brown*	W13 **Deciding Between Relational Databases and Tables in the Cloud** *Michael Stiefel*	W14 **BI in the Cloud with SQL Azure Reporting** *Eric D. Boyd*
W15 **HTML5 and Internet Explorer 9: Developer Overview** *Ben Hoelting*	W16 **Radically Advanced Templates for WPF and Silverlight** *Billy Hollis*	W17 **Windows Azure Platform Overview** *Vishwas Lele*	W18 **Overview of Project 'Crescent'** *Andrew Brust*

Wild Wednesday with Developer Duel

Visual Studio Live! Day 3: Thursday, December 8, 2011

TH1 **Getting Started with ASP.NET MVC** *Philip Japikse*	TH2 **XNA Games for Windows Phone 7** *Brian Peek*	TH3 **Building Windows Azure Applications** *Vishwas Lele*	TH4 **REST with Silverlight 5, WCF Web API, and a Little ASP.NET MVC3** *Pete Brown*
TH5 **Test Driving ASP.NET MVC** *Philip Japikse*	TH6 **Working with Data on Windows Phone 7** *Sergey Barskiy*	TH7 **Building Compute-Intensive Apps in Windows Azure** *Vishwas Lele*	TH8 **Session** *TBA*
TH9 **Busy Developer's Guide to (ECMA/Java)Script** *Ted Neward*	TH10 **Building Native Mobile Apps with HTML5 & jQuery** *Jon Flanders*	TH11 **Building and Running the Windows Azure Developer Portal** *Chris Mullins*	TH12 **Multi-Touch Madness!** *Brian Peek*

Lunch

TH13 **Using Code First (Code Only) Approach with the Entity Framework** *Sergey Barskiy*	TH14 **Getting Started with Windows Phone 7** *Scott Golightly*	TH15 **The LINQ Programming Model** *Marcel de Vries*	TH16 **Application Lifecycle Management and Visual Studio: What's Next** *Brian Randell*
TH17 **Using MEF to Develop Composable Applications** *Ben Hoelting*	TH18 **Data Binding and MVVM Patterns in HTML5** *John Papa*	TH19 **Microsoft Session** *TBA*	TH20 **Visual Studio v.Next** *Brian Randell*

Visual Studio Live! Post-Conference Workshops: Friday, December 9, 2011 *(Separate entry fee required)*

FWK1 **Workshop: Architectural Katas** *Ted Neward*	FWK2 **Workshop: ALM in 2011: Visual Studio 2010 and the Next Big Release** *Brian Randell*

For the complete session schedule and full session descriptions, please check the Visual Studio Live! Orlando web site at **vslive.com/Orlando**
**Speakers and Sessions Subject to Change.*

VISIT US ONLINE AT VSLIVE.COM/ORLANDO FOR MORE DETAILS!

Custom Claims-Based Security in SharePoint 2010

Ivory Feng, Patrick Stanko and Shabbir Darugar

Microsoft SharePoint 2010 introduced a new claims-based identity model for building claims-aware applications. When a user comes to a claims-aware application, she presents an identity to the application as a set of claims. These claims enable more granular authorization to content. Thus, users with certain attributes, such as Country Code and Department Code, are granted access to the appropriate content.

Implementing claims-based security at the enterprise level presents some interesting challenges. We'll explore the process of building a custom claims provider (CCP) in SharePoint, integrating this with Microsoft FAST Search Server 2010 for SharePoint, managing claims-enabled content from a user's perspective and avoiding certain gotchas. As we do this, you'll learn how to design and implement a dynamic custom claims-based security model in SharePoint 2010. We'll use the Contoso company's human resource portal, ContentWeb, for our example.

Overview of the Application

Contoso is a global company with close to 100,000 employees, and ContentWeb holds a wealth of information that content authors need to manage on a daily to weekly basis. In most cases, the content is applicable to folks in one country but not another, to one discipline but not another, or to all Contoso employees. Some of the content, however, is sensitive data that needs to be available to only a small subset of employees (such as VPs).

To cover such a broad audience, ContentWeb uses SharePoint Groups (SPGs) with custom claims to explicitly secure and target content. Claims specific to Contoso's needs are stored in a database and used to define a user's claim "set." It is this set of claims that determines what an employee can and can't see. An administrative interface lets the information workers that support the portal change or augment their own claims set, giving them the ability to view the portal as a different user. This powerful custom feature is called View Portal As (VPA).

ContentWeb uses claims-based Windows authentication in SharePoint 2010 and FAST Search Server 2010 for SharePoint as the search engine in a federated enterprise search environment.

This article discusses:

- The new claims-based identity model in Microsoft SharePoint Server 2010
- Why build a custom claims provider
- Simple and compound claims
- Making the database connection string available to Web front ends
- Using SharePoint Groups
- Implementing search using Microsoft FAST Search Server 2010 for SharePoint

Technologies discussed:

Microsoft SharePoint Server 2010

Code download available at:

code.msdn.microsoft.com/mag201111SPSecurity

Why a CCP?

When a user logs into ContentWeb, the claims-based Windows authentication provides a set of claims from Active Directory, which acts as the identity provider. However, these claims alone are not sufficient for ContentWeb to secure and target content. ContentWeb requires additional user information from enterprise data repositories such as SAP to build a full claim set that describes the user. Thus, we need a CCP to augment the claims. We use these custom claims to explicitly secure and target content to the user.

Having a CCP allows Contoso to determine *what* claim types and values it wants to use to describe employees, vendors and others it works with. This model is scalable; it allows Contoso to efficiently change over time as its business changes.

Because you can't easily edit the claims coming from Active Directory, you also need a CCP to support VPA. VPA is primarily used when an authorized person (User A) wants to view the portal as another person (User B). In such cases, the custom claims provider adds User B's custom claims into User A's claim set, thus authorizing User A to view content that User B can access.

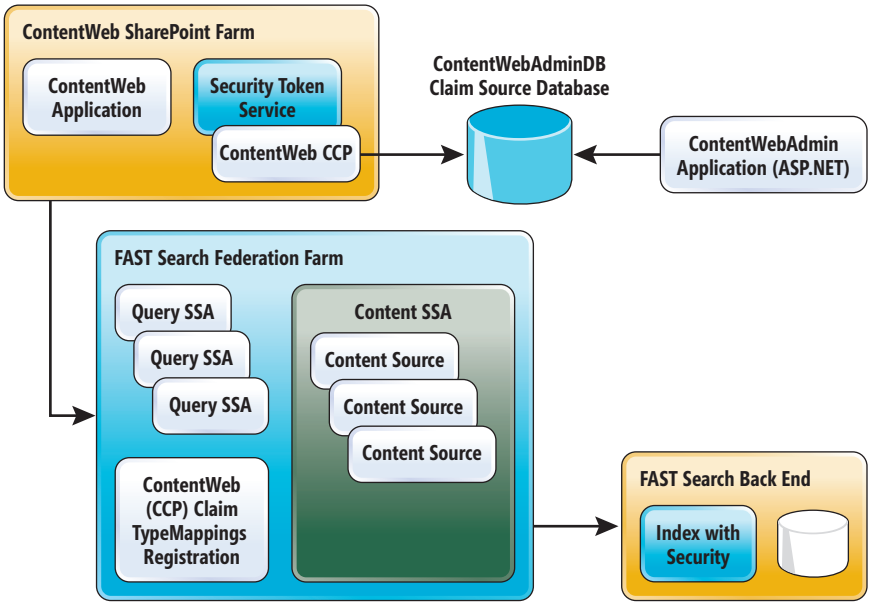


Figure 1 **A High-Level View of ContentWeb**

The claims a user possesses are attributes that represent his "profile." Users can have both simple and compound claims. Simple claims, such as CountryCode and UserType, are easy to understand. ContentWeb uses a predefined set of simple claims—or, more precisely, simple claim types. However, sometimes an employee needs to be granted permission to content that has more than one claim type; for example, CountryCode=US AND UserType=FullTimeEmployee—the AND condition. In that case, you create compound claims.

Out of the box, SharePoint 2010 evaluates claims in a user's claim set using OR logic only. This means SharePoint 2010 supports only the OR condition for security and targeting, and that's not always sufficient. Let's look at an example. ContentWeb has both a CountryCode claim and a UserType claim. It needs to secure some content for full-time U.S. employees using CountryCode = US AND UserType = FullTimeEmployee. We can't satisfy this requirement directly in SharePoint 2010 using the simple claims CountryCode or UserType, so we need to create a compound claim to represent using both claims.

We could create a new claim type for full-time U.S. employees. However, if the company creates new offices in other countries, we then need to update our CCP source code to add corresponding claim types. Clearly, this approach can be costly and time-consuming. Moreover, the ContentWeb business team wants to be able to add and remove compound claim types without needing a new application release each time. As you can see, ContentWeb requires a dynamic and extensible security model.

How will we manage all of the claims data? We'll use a database called ContentWebAdminDB to store claim metadata. ContentWebAdminDB is the claims source back end; it consolidates all the claims source data from SAP and other enterprise databases. We'll delve into this database more in a bit. And we'll use an ordinary ASP.NET administrative application called ContentWebAdmin to connect to this database to create and update compound claims metadata and VPA configuration. (Details of this application are out of scope for this article.) Compound claims are created using simple claims. For example, in the ContentWebAdmin application we can create a compound claim called US-FullTimeEmployee using the CountryCode = US and UserType = FullTimeEmployee simple claims. With this coumpound claim metadata, for any user who has CountryCode = US and UserType = FullTimeEmployee simple claims, a new compound claim US-FullTimeEmployee = true will be added into the user's claim set. Note that no compound claim US-FullTimeEmployee = false appears in any user's claim set. With this dynamic compound claims design, each user has only a few dozen claims in total even though the permutation of compound claims can be pretty high. This is the key to ensuring good performance.

When a user signs in to ContentWeb, the CCP is invoked. The CCP calls a custom stored procedure, sp_GetUserClaims, in ContentWebAdminDB while passing in the user's identity claim. The stored procedure checks VPA configuration and returns all the simple and compound claims for that user to the CCP. As you can see, dynamic compound claims are not the same as dynamic authorization rules. Our compound claims solution doesn't change the way SharePoint 2010 evaluates claims. We simply inject compound claims into user claim sets based on the compound claims metadata configuration. **Figure 1** shows a high-level overview of ContentWeb.

Inside the ContentWeb Custom Claims Provider

The ContentWeb CCP has a special class that inherits from Microsoft.SharePoint.Administration.SPClaimProvider. **Figure 2**

Figure 2 **Making Claims More Readable**

```
protected override void FillResolve(
  Uri context, string[] entityTypes, SPClaim resolveInput,
  List<Microsoft.SharePoint.WebControls.PickerEntity> resolved)
{
  string claimTypeName = string.Empty;
  string claimValue = string.Empty;

  // Handles only ContentWeb claim types
  if (null != resolveInput && resolveInput.ClaimType.Contains(
    ContentWebClaimTypes.ContentWebClaimTypePrefix))
  {
    claimValue = resolveInput.Value.ToLower();
    claimTypeName = GetName(resolveInput.ClaimType);

    PickerEntity entity = CreatePickerEntity();
    entity.Claim = resolveInput;
    entity.Description = resolveInput.ClaimType + " = " + claimValue;
    entity.DisplayText = claimTypeName + " = " + claimValue;
    entity.EntityData[PeopleEditorEntityDataKeys.DisplayName] =
      claimTypeName + " = " + claimValue;
    entity.EntityType = SPClaimEntityTypes.User;
    entity.IsResolved = true;
    resolved.Add(entity);
  }
}
```

Figure 3 **The ConnectionStringStorage Class**

```
[Guid("56705e15-abd3-44f0-adea-91488da1a572")]
public class ConnectionStringStorage
  : Microsoft.SharePoint.Administration.SPPersistedObject
{
    [Persisted]
    private string m_connectionString;

    public ConnectionStringStorage()
    {
    }

    public ConnectionStringStorage(
      string name, SPPersistedObject parent)
      : base(name, parent)
    {
    }

    public ConnectionStringStorage(
      string name, SPPersistedObject parent, Guid guid)
      : base(name, parent, guid)
    {
    }

    public string ConnectionString
    {
      get { return m_connectionString; }
      set { m_connectionString = value; }
    }
}
```

shows part of this class. Please note the highlighted line of code. We added it to help make our custom claims types and values display in a more user-friendly and readable manner.

The ContentWeb CCP needs a connection string to call a stored procedure in ContentWebAdminDB for user claims. Because the CCP is a farm-level feature, it's not within the scope of any Web application, so we can't use the web.config file to configure the database connection string. We could use machine.config, but that's not ideal because we have multiple Web front ends. Instead, we'll use the SharePoint hierarchical object store, SPPersistedObject, to store the database connection string. This is stored in the SharePoint configuration database, making the database connection string available to all Web front ends in the SharePoint farm.

Inside the ContentWeb CCP, we create a class that inherits from Microsoft.SharePoint.Administration.SPPersistedObject, as shown in **Figure 3**.

We'll use the following Windows PowerShell script to register the database connection string:

```
[System.Reflection.Assembly]::LoadWithPartialName(
  "Microsoft.Sample.ContentWeb.ClaimsSecurity")
$id = new-object System.Guid("56705e15-abd3-44f0-adea-91488da1a572")
$farm = Get-SPFarm
$existingObject = $farm.GetObject($id)
$existingObject.Delete()
$newObject =
  new-object Microsoft.Sample.Contoso.ClaimsSecurity.ConnectionStringStorage(
  "ConnectionString", $farm, $id)
$newObject.ConnectionString = "Data Source=ContentWebAdminSQLServer;
  Initial Catalog=ContentWebAdminDB;Integrated Security=True;Timeout=30"
$newObject.Update();
Iisreset
```

As you can see, we reference the CCP dll. We create a new ConnectionStringStorage object and set its ConnectionString property. Finally, we call its Update method to save it into the SharePoint config database.

We recommend the connection string have a timeout value of less than 60 seconds. The CCP has an execution default timeout of 60 seconds. This means that if the database connection times out, you have the ability to capture the real exception from the connection string instead of a misleading one from the CCP execution.

Inside the CCP's constructor, we'll retrieve the connection string and store it in a module-level variable:

```
public class ContentWebClaimProvider : SPClaimProvider
{
  private static string connectionString = null;

  public ContentWebClaimProvider(string displayName)
    : base(displayName)
  {
    Guid guid = new Guid(@"56705e15-abd3-44f0-adea-91488da1a572");
    ConnectionStringStorage storage =
      (ConnectionStringStorage)SPFarm.Local.GetObject(guid);
    connectionString = storage.ConnectionString;
  }
}
```

We want to be able to trace when the CCP connects to the database for user claims. SharePoint 2010 has a new feature that's perfect for this. It's called the Microsoft.SharePoint.Utilities.SPMonitoredScope object, and it monitors performance and resource use for a specified block of code. The following code logs the CCP's database calls:

```
using (new SPMonitoredScope("ContentWeb.CCP.GetUserClaims", 5))
{
  userClaimsDataSet = CustomClaimSourceDA.GetUserClaims(
    userAlias, connectionString);
}
```

We can also configure the event level and trace level of this event in SharePoint Central Administration. These are controlled by the Monitoring object in the SharePoint Foundation category under Monitoring | Reporting | Configure diagnostic logging.

Inside the ContentWebAdminDB Database

As noted earlier, ContentWebAdminDB consolidates the user's claim data from SAP and other enterprise databases, and it contains the custom claim source. The ContentWeb CCP calls a stored procedure, sp_GetUserClaims, to retrieve a user's custom claims.

The ContentWebAdmin application allows authorized users to configure compound claims and VPA for a user. ContentWebAdminDB stores the compound claims configuration metadata and the VPA configuration in tables. When the ContentWeb CCP calls sp_GetUserClaims, this stored procedure will check VPA configuration and then return both simple claims and compound claims. The detailed design and implementation of this database is out of the scope of this article. However, we included a SQL script for a dummy sp_GetUserClaims stored procedure with hardcoded claim values within the downloadable code package.

Using SharePoint Groups

An SPG is a logical wrapper of SharePoint users. This is handy for ContentWeb, which uses hundreds of compound claims. Without SPGs, it would be very difficult for content authors to manage all of the claims.

Using SPGs provides the following benefits:

- **Friendly Naming:** SPGs let users create friendly and meaningful group names for potentially cryptic claim values. For example, ContentWeb has an SPG named "US FullTime Employees" that describes users who have the claim values Country=United States, CompanyCode=1010 and UserSubType=FTE.
- **People Picker Name Resolution Support:** The People Picker control in SharePoint 2010 natively supports name

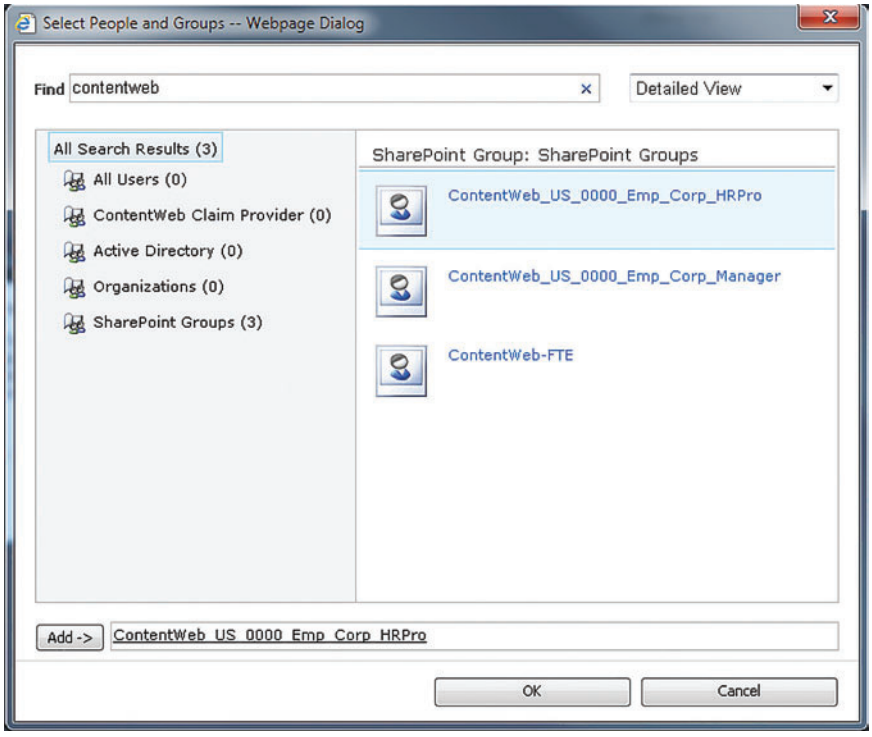


Figure 4 **SharePoint Groups in People Picker**

resolution for SPGs. This makes it easier to find and select SPGs, as shown in **Figure 4**.

- **Dynamic Claim Security Management:** Suppose you needed to change the definition of "US FullTime Employees," adding UserType=Corp to the other attributes—Country=United States, CompanyCode=1010 and UserSubType=FTE—due to changing business needs. It's easy to make the change in just that one SPG, and all objects secured in that SPG would automatically inherit this change.
- **Just One SPG for Securing and Targeting Content:** SharePoint 2010 now supports the ability to secure and target an object using the same SPG. This eliminates the typical overhead associated with creating audiences, importing user profiles, running the profile sync job and so forth for targeting.
- **Multiple Claims Associated with One SPG:** Let's say you need to secure a page for full-time employees whose level ranges from 60 to 65. You can easily do this by creating an SPG that has compund claims for UserSubType=FTE and Level=60, UserSubType=FTE and Level=61… UserSubType=FTE and Level=65. SharePoint 2010 interprets all of these claims as "OR."

Managing many SPGs has the potential to be time-consuming, cumbersome and error-prone. The ContentWeb business team uses an Excel file to manage the hundreds of groups with associated custom claims.

There are various options for automating the creation of SPGs. We created a Windows PowerShell script that reads data from an XML file to automate this. (You'll find this script in the code download, which can be downloaded from code.msdn.microsoft.com/mag201111SPSecurity.) We populated the XML data file using the business team's Excel file. **Figure 5** shows the sample XML configuration file.

Using SPGs to Target

To use targeting, you need the User Profile service application enabled for the Web app.

There's one tricky issue to watch out for when using SPGs for targeting. We initially found that the SPGs containing Active Directory users or security groups were all working fine, but the SPGs containing custom claim users didn't work at all. The root cause was related to the permission level of those groups. When we created the SPGs with custom claims, we didn't want to assign any default permission level. However, if an SPG is not assigned a permission level when created and later used for security, it will be flagged with a special permission level called Limited Access, as shown in **Figure 6**.

Because ContentWeb has hundreds of SPGs and most of them are used only for the target audience, they don't have any permission level set. However, the target audience processor can't pull in these SPGs without any permission level. We were lucky and discovered a workaround via the permission UI of the page we were applying to the target audience. We granted "Read" permission to the SPGs being used for the target audience, then immediately undid this permission. This process causes SharePoint to flag these SPGs with the Limited Access permission level, and the target audience happily honors that permission level. It's important to note that you don't have to go through this grant and undo process for the SPGs multiple times. You need to do it just once within the security inheritance tree. Be aware that if you break security inheritance, you'll need to do it again, even if you've done it at the parent level.

Implementing Search for ContentWeb

One of our major goals with this application was to improve the search experience for Contoso employees. We wanted ContentWeb to make content far more discoverable than it had been in the past.

To do this for ContentWeb, we used FAST Search Server 2010 for SharePoint hosted in a federated environment. Refer back to **Figure 1** for Contoso's federated FAST Search environment. This

Figure 5 **Sample XML Configuration File**

```
<?xml version="1.0" encoding="utf-8"?>
<SharePointGroups url="http://contentweb" owner="contoso\ivfeng" >
  <SharePointGroup name="ContentWeb-FTE"
    description="ContentWeb-FTE" permissionLevel="Read">
    <Claim type="UserType" value="emp"/>
  </SharePointGroup>
  <SharePointGroup name="ContentWeb_US_0000_Emp_Corp_HRPro"
    description="ContentWeb_US_0000_Emp_Corp_HRPro">
    <Claim type="CompoundClaim" value="US+emp+corp+hrpro"/>
  </SharePointGroup>
  <SharePointGroup name="ContentWeb_US_0000_Emp_Corp_Manager"
     description="ContentWeb_US_0000_Emp_Corp_Manager">
    <Claim type="CompoundClaim" value="us+emp+corp+manager"/>
    <Claim type="CompoundClaim" value="US+emp+corp+hrpro"/>
  </SharePointGroup>
</SharePointGroups>
```

Figure 6 **A SharePoint Group with Limited Access**

shows a SharePoint farm called the FAST Search Federation Farm. Behind this farm is the Microsoft FAST Search Server for SharePoint engine and database. The FAST Search farm contains all the Query Search Service Applications (Query SSAs) and a Content SSA.

Naturally, we created a content source within the existing Content SSA. (FAST Search Server recommends only one Content SSA.) Once this is done, the Content SSA will crawl the ContentWeb content database and make the search index available to ContentWeb and other intranet applications.

We then created a separate Query SSA and published it to ContentWeb. With the initial search test on ContentWeb, we found that basic search worked, but search security trimming wasn't working as expected. For any pages secured with ContentWeb custom claims, users could not see the links within the search results even though they had the security permissions. We finally figured out that we needed to have the ContentWeb custom claim type mappings registered in the FAST Search SharePoint farm for search security trimming to work. The simplest way to do this is to deploy the CCP into the FAST Search SharePoint farm. Not only did we have to confirm the claim type mappings were registered, we also had to make sure all claim IDs (shown as Account in **Figure 7**) were the same across all the SharePoint farms for the same claim type value pair.

Currently, even with the SharePoint 2010 June 2011 cumulative updates, claim type mappings in SharePoint 2010 are immutable. Once a CCP is deployed on a SharePoint 2010 farm, the claim type mappings are registered and remain unchanged even if you remove and redeploy the CCP. Because you may have CCPs on multiple SharePoint farms, it's very important that you deploy all the CCPs in the same sequence to ensure the claim IDs are the same across all farms. If you don't do this, the sequence IDs won't be aligned and search won't return accurate results.

When ContentWeb content is crawled, the CCP is not invoked. All that's needed is the claim type mappings registered in the FAST Search farm so that the crawler engine can decode the ContentWeb custom claims. It then stores the decoded custom claims for security in the search index. Removing the ContentWeb CCP is optional because the claim type mappings will remain. Moreover, we don't really need the ContentWeb CCP to augment user claims on the FAST Search farm.

Security Trimming Search Results: Contoso Intranet Portals

In order for the ContentWeb content to be searchable from other Contoso intranet portals, you need to ensure the following:

- All intranet portals that need to surface the ContentWeb content must be built on SharePoint 2010 and be claims-based Web applications that use Windows authentication.
- All intranet portals must query the same FAST Search index.
- All intranet portals must have the ContentWeb CCP installed on them. (If there's more than one CCP, they all must be installed in the same sequence to ensure claim IDs are aligned.)

If all of these conditions are met, the ContentWeb content will be searchable across all of the Contoso intranet portals.

> Custom claim type mappings need to be registered in the FAST Search SharePoint farm.

Wrapping Up

SharePoint 2010 provides strong integration with claims-based security. SharePoint groups, claims and a custom claims provider allow you to authorize content for the enterprise broadly as well as at a granular level. As you can see, you can satisfy many scenarios with relative ease.

This design also lets developers build robust tools that allow information workers (such as support personnel) to alter their claims and view a site as a different user, and then revert.

As with most new design implementations, our experience didn't come without "opportunities to learn," but the end product was well worth it. The sample code we've provided should help get you going in the right direction and off to a great start on deploying your next-gen enterprise portal! ■

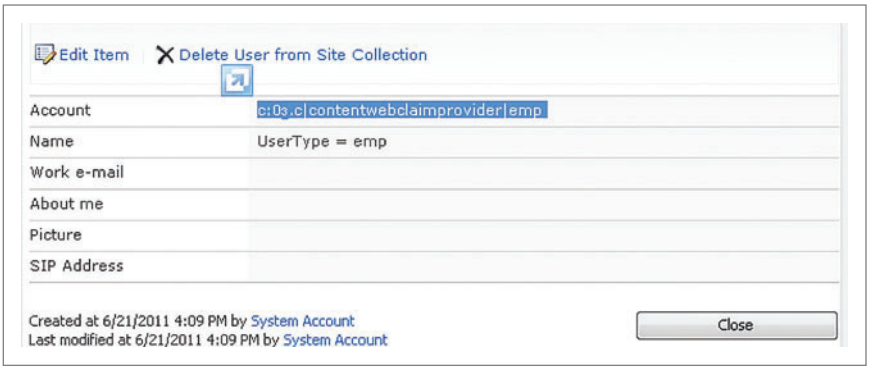


Figure 7 **Claim IDs**

JINHUI (IVORY) FENG *is a senior software development engineer at Microsoft working on its internal human resources applications.*

SHABBIR DARUGAR *is a senior software development engineer and project lead at Microsoft working on its internal human resources applications.*

PATRICK STANKO *is a lead program manager at Microsoft working on its internal human resources applications.*

THANKS to the following technical expert for reviewing this article: *Tom Wisnowski*

Developing 3D Objects in Silverlight

Rajesh Lal

In this article, I'll show you how to develop 3D objects in Silverlight. I'll start with a brief background on 3D and then look at some of the advanced features available in Silverlight that enable creation and display of 3D objects. I'll take a simple example of a cube and show you three different ways to create 3D transformations. I'll also explain what key elements you need to display a 3D object on a computer screen. Finally, I'll explore how Silverlight 5 will allow you to go beyond what's available today and create much richer 3D objects.

Silverlight supports a right-handed coordinate system, which means the positive z axis is pointing toward the viewer (see **Figure 1**). There are three main elements of 3D that are needed for displaying an object on the screen:

- Perspective
- Transformation
- Effect of light

Perspective means that parts of objects closer to us appear larger than those that are farther away. For example, in **Figure 1**, side *bd* looks bigger than side *fh*. In real life, perspective creates a vanishing point, which means if you extend the lines *ae*, *bf*, *cg* and *dh* on the z axis, they'll meet far back at an arbitrary single point.

The second aspect is transformation. A 3D object, when displayed on a screen, should allow movement in the 3D space in all directions. It can be moved in any single axis—scaled for size—while keeping the perspective intact. It can be rotated 360 degrees in all axes: x, y and z. This gives a 3D object the flexibility it needs for rendering on the screen.

The last element of 3D is the effect of light. Light in 3D creates shading, which is brighter near a light source and fades with distance. In 3D rendering, the two popular kinds of shading are "flat" shading and "gradient" shading. I'll explain how they differ later. Light will also create a shadow on the sides opposite the light source.

In the forthcoming examples, I'll explore three different ways in which you can create 3D objects in Silverlight:

- Using perspective 3D
- Using multiple frames and a timer
- Using primitives with the XNA library

In the first method, an object is created using two-dimensional elements, but it looks and behaves as if it were in 3D space. Perspective

This article discusses a prerelease version of Silverlight 5. All related information is subject to change.

This article discusses:

- Perspective 3D
- Frames 3D
- Mapping 3D objects to the screen
- The Silverlight 5 matrix struct
- The XNA 3D pipeline
- Using XNA primitives

Technologies discussed:

Silverlight

Code download available at:

code.msdn.microsoft.com/mag201111Silverlight

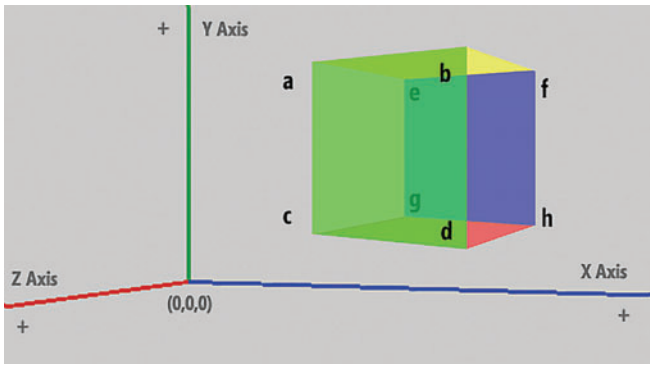


Figure 1 **A Reference Cube, Showing Sides with Perspective**

3D is a special kind of transformation feature added in Silverlight 4 that allows for basic transformations such as rotation, scaling and translation in 3D space. The second method is the brute force method, where you don't create the 3D object, but instead create the final output frames for a particular transformation and display them using a timer. The final method is to create a rich 3D object bit by bit using primitives (a list of triangles) with XNA libraries, which will be available in Silverlight 5. Let's get started.

Creating a Cube Using Perspective 3D

Silverlight 4 supports a PlaneProjection class (see **Figure 2**), which can be used for the Projection property of any UI element, as shown in the class diagram. The PlaneProjection class allows for perspective 3D transformations on the UI element. Although it doesn't directly allow for creating a 3D object, you can use multiple "walls" to create an object and transform it in 3D space.

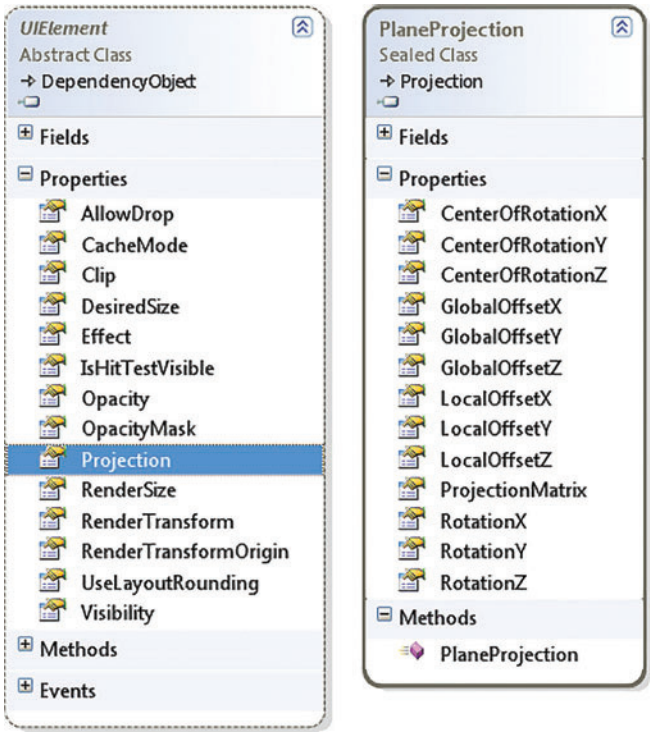


Figure 2 **The PlaneProjection Class**

The PlaneProjection class allows for LocalOffset and GlobalOffset, which are used to translate an object with respect to itself and with respect to another element in the global space, respectively. RotationX, RotationY and RotationZ allow for rotating the element in the x, y and z axes, and CenterOfRotation allows for a center point with respect to the element plane.

In my example, to create a "cube," I'll create four sides of the cube and move them in the 3D space by setting their PlaneProjection properties, as shown in **Figure 3**.

In **Figure 4**, the sides are rotated by 90, -90 and -180 degrees to create the top, bottom and front projection planes of the cube. The CenterofRotationZ value of 125 creates the center point by which all the planes can be rotated along the z axis.

Once the cube is created using the projection plane, I need to rotate it by the x, y and z axes. This is where I use the storyboard object in Silverlight. I create three storyboards, one for each axis, as shown in **Figure 5**.

> I was easily able to create a cube and transform it in 3D space without a lot of code.

On each storyboard, I rotate each of the four projection planes to keep the cube structure intact. Note that for x axis rotation, the RotationX value starts from the original RotationX value of the plane and goes another 360 degrees for ProjectionFront, so it starts at -180 degrees and goes to 180 degrees. As you can see in **Figure 6**, the cube is ready for rotation along the x, y, and z axes, it can be moved along any axis and it supports coloring of each of the sides.

In this example, I was easily able to create a cube and transform it in 3D space without a lot of code. This is the real strength of perspective 3D transformation. For basic 3D operations, you should

Figure 3 **Setting PlaneProjection Properties**

```
<Grid x:Name="LayoutRoot" Background="White" Width="800" Height="700">
  <Rectangle Fill="#9900FF00" Width="250" Height="250" Visibility="Visible">
  <Rectangle.Projection>
    <PlaneProjection x:Name=
      "projectionFront" CenterOfRotationZ="125" RotationX="-180"/>
  </Rectangle.Projection>
</Rectangle>
<Rectangle Fill="#99FF0000" Width="250" Height="250" Visibility="Visible">
  <Rectangle.Projection>
    <PlaneProjection x:Name=
      "projectionBottom" CenterOfRotationZ="125" RotationX="-90" />
  </Rectangle.Projection>
</Rectangle>
<Rectangle Fill="#990000FF" Width="250" Height="250" Visibility="Visible">
  <Rectangle.Projection>
    <PlaneProjection x:Name="projectionBack" CenterOfRotationZ="125" />
  </Rectangle.Projection>
 </Rectangle>
<Rectangle Fill="#99FFFF00" Width="250" Height="250" Visibility="Visible">
  <Rectangle.Projection>
    <PlaneProjection x:Name=
      "projectionTop" CenterOfRotationZ="125" RotationX="90"/>
  </Rectangle.Projection>
</Rectangle>
</Grid>
```

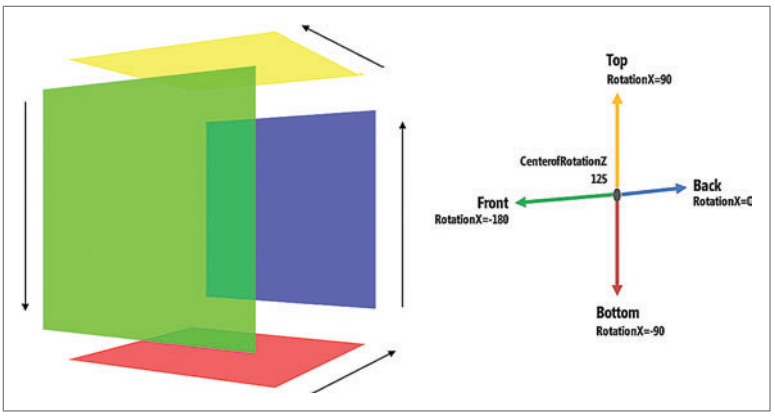


Figure 4 **Projecting the Sides to Simulate a 3D Wall**

use this option. However, it comes with a number of drawbacks. For advanced 3D objects, the number of projection planes required and their settings can increase dramatically, and you have to manually figure out the angles between each projection plane and the CenterOfRotation. A second issue here is that rotation of the 3D object depends on the storyboards, which makes it CPU-intensive; it doesn't use the GPU to render the object. Another issue with this approach is that you also render the back of the cube even if it isn't visible—not an optimal approach.

Figure 5 **Rotating the Cube with Storyboards**

```
<Storyboard x:Name="storyboardRotateX">
  <DoubleAnimation Storyboard.TargetName="projectionFront"
    Storyboard.TargetProperty="RotationX" From="-180.0" To="180.0" Duration="0:0:10"
    RepeatBehavior="Forever"  />
  <DoubleAnimation Storyboard.TargetName="projectionBottom"
    Storyboard.TargetProperty="RotationX" From="-90.0" To="270.0" Duration="0:0:10"
    RepeatBehavior="Forever"  />
  <DoubleAnimation Storyboard.TargetName="projectionBack"
    Storyboard.TargetProperty="RotationX" From="0.0" To="360.0" Duration="0:0:10"
    RepeatBehavior="Forever"   />
  <DoubleAnimation Storyboard.TargetName="projectionTop"
    Storyboard.TargetProperty="RotationX" From="90.0" To="450.0" Duration="0:0:10"
    RepeatBehavior="Forever"   />
  </Storyboard>
  <Storyboard x:Name="storyboardRotateY">
  <DoubleAnimation Storyboard.TargetName="projectionFront"
    Storyboard.TargetProperty="RotationY" From="0.0" To="360.0" Duration="0:0:10"
    RepeatBehavior="Forever" />
  <DoubleAnimation Storyboard.TargetName="projectionBottom"
    Storyboard.TargetProperty="RotationY" From="0.0" To="360.0" Duration="0:0:10"
    RepeatBehavior="Forever" />
  <DoubleAnimation Storyboard.TargetName="projectionBack"
    Storyboard.TargetProperty="RotationY" From="0.0" To="360.0" Duration="0:0:10"
    RepeatBehavior="Forever" />
  <DoubleAnimation Storyboard.TargetName="projectionTop"
    Storyboard.TargetProperty="RotationY" From="0.0" To="360.0" Duration="0:0:10"
    RepeatBehavior="Forever" />
  </Storyboard>
  <Storyboard x:Name="storyboardRotateZ">
  <DoubleAnimation Storyboard.TargetName="projectionFront"
    Storyboard.TargetProperty="RotationZ" From="0.0" To="360.0" Duration="0:0:10"
    RepeatBehavior="Forever" />
  <DoubleAnimation Storyboard.TargetName="projectionBottom"
    Storyboard.TargetProperty="RotationZ" From="0.0" To="360.0" Duration="0:0:10"
    RepeatBehavior="Forever" />
  <DoubleAnimation Storyboard.TargetName="projectionBack"
    Storyboard.TargetProperty="RotationZ" From="0.0" To="360.0" Duration="0:0:10"
    RepeatBehavior="Forever" />
  <DoubleAnimation Storyboard.TargetName="projectionTop"
    Storyboard.TargetProperty="RotationZ" From="0.0" To="360.0" Duration="0:0:10"
    RepeatBehavior="Forever" />
</Storyboard>
```

The third main element needed for displaying 3D objects onscreen is the effect of light. In real life, you have light, so how do you simulate that in 3D space on the screen? As mentioned, two popular ways to do that are flat shading and gradient shading.

Flat shading takes into account the surface of the plane and applies an average shading along the plane. Gradient shading (Gouraud shading) uses a gradient to shade the surface and takes into account each of the vertices of the plane. The plane isn't flat shaded, but rather "smooth" shaded, based on different vertex colors.

In this example, each of the planes allows for color fill (flat shading) as well as gradient fill (gradient shading), which can be used to simulate the light effect. I'll discuss these later. A quick way to simulate the light effect is by overlapping a Radial gradient rectangle with transparency using the following code after the 3D object:

```
<Rectangle x:Name=
  "BulbGradient" Height="700" Width="800" Margin="0 50 0 0" Grid.Row="1"
  Visibility="Collapsed">
  <Rectangle.Fill>
    <RadialGradientBrush RadiusX="0.5" RadiusY="0.5" GradientOrigin="0.25,0.25">
      <GradientStop Color="#00000000" Offset="0"/>
      <GradientStop Color="#FF000000" Offset="2"/>
    </RadialGradientBrush>
  </Rectangle.Fill>
</Rectangle>
```

Creating a Cube Using Frames

The second way to create a 3D experience is by using the final frames. Here, you don't create the 3D object itself, but start with the required final output and export it as individual frames. A number of 3D modeling software programs allow you to create 3D objects and transformations that can be exported as multiple frames and then imported into Silverlight.

In this example, I'll take a simple cube animation and export its rotation on the x, y and z axes into multiple frames of images. **Figure 7** shows eight different frames of a cube rotation on the x axis. For this example, I use a minimal set of frames to create a cube rotation, but more frames per second create a smoother and seamless rotation.

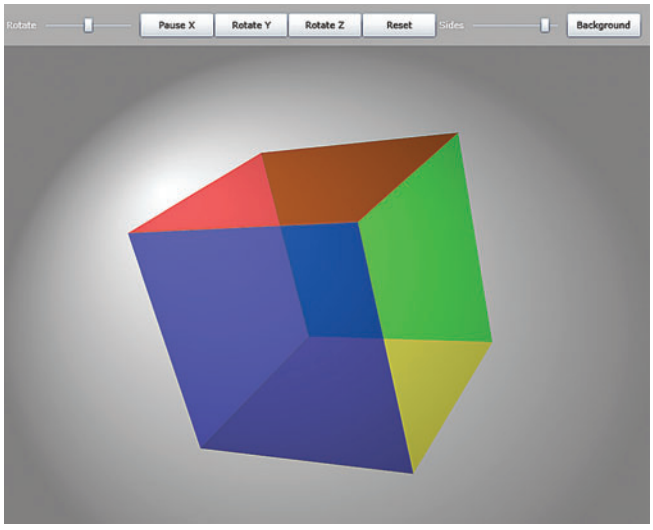


Figure 6 **The Cube, Ready for Rotation**

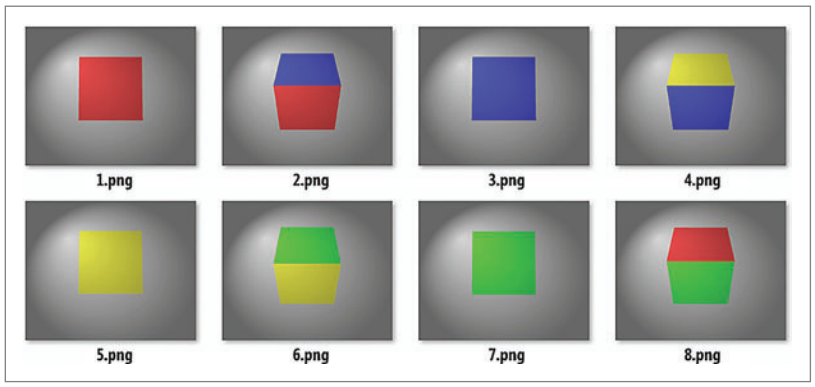


Figure 7 **Eight Different Frames Simulating Rotation in the X Axis**

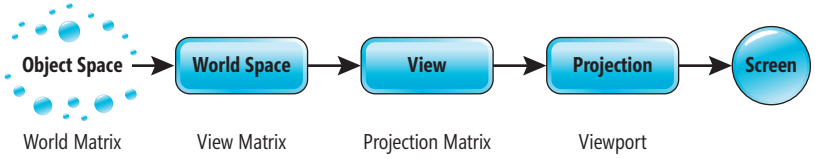


Figure 8 **The Order in Which a 3D Object Is Mapped to the Screen**

To simulate the rotation in Silverlight, I use a timer as shown in the following code:

```
DispatcherTimer timer = new DispatcherTimer();
timer.Interval = new TimeSpan(0, 0, 0, 0, 500);
timer.Tick += new EventHandler(Tick);

private void Tick(object o, EventArgs sender)
{
  string imageuri = "cube/" + axis + "/" + currentImageIndex + ".png";
  bgImage.Source = new BitmapImage(new Uri(imageuri, UriKind.RelativeOrAbsolute));
  if (currentImageIndex <= 8)
    currentImageIndex++;
  else
    currentImageIndex = 1;
}
```

Note that this is a simplified version of what you can do with this method. Here I'm using exported images for simplicity, but a number of 3D softwares allow for XAML export, which creates polygons with color and gradients instead of images, providing a seamless, smoother animation. The timer can also be replaced by storyboards with animations.

This approach is straightforward. You have a specific transformation requirement with a particular 3D object. You don't have to worry about creating the 3D object, and this technique can be used to create any kind of 3D object—you're no longer restricted to a simple cube. This method can be used for all kinds of transformations. You can translate, rotate and scale sophisticated 3D objects. It even simulates the effect of light, which also translates directly from the modeling software.

The primary limitation of this approach is flexibility of programming against the 3D object. Once you export your 3D object, it generates static code that can be difficult to code against. You can't move the object with respect to other elements in your Silverlight application. Another disadvantage is that the number of frames required increases linearly with every transformation. Apart from that, the rendering happens on the CPU, so bigger animations with larger numbers of frames would have a performance hit.

This leads to the third approach of using the XNA library in the upcoming Silverlight 5, which, as you'll see, overcomes most of the problems with the first two approaches. But before that, let's talk about how a 3D object translates in a 2D screen mathematically—the genius logic behind 3D.

Understanding World, View and Projection Matrix

To display the object, you have to understand three main concepts or "spaces" and how the object is mapped from its own object space to the screen:

- World
- View
- Projection

Figure 8 shows the order in which an object is mapped to the screen.

The first set of coordinates for a 3D object is the x, y, and z coordinates in the object space (also called model space). These coordinates are relative to one another with their centers (0, 0, 0). Remember, with right-hand Cartesian coordinates, the positive z axis is toward the viewer.

For a 3D cube, the top-right corner of the front side will be (1,1,1), and the bottom-left corner of the back side will be (-1,-1,-1), as shown in **Figure 9**. In object space, coordinates are relative to one another and their position can only range from -1 to +1. For my cube to use 75 percent of the object space, I need to multiply each coordinate by .75, so the new *b* will be (.75,.75,.75) and the new *g* will be (-.75,-.75,-.75).

When the object is put on the world space, the object itself isn't moved, but rather mapped with respect to the world coordinates by multiplying its coordinates with the world matrix. In the world space, you can transform the 3D object by moving the coordinates to translate the object, changing the size to scale and changing the angle to rotate the object. To express your object's coordinates in the world space, you need to multiply each vertex position with the world matrix:

Object World Coordinates = Object Coordinates * World Matrix

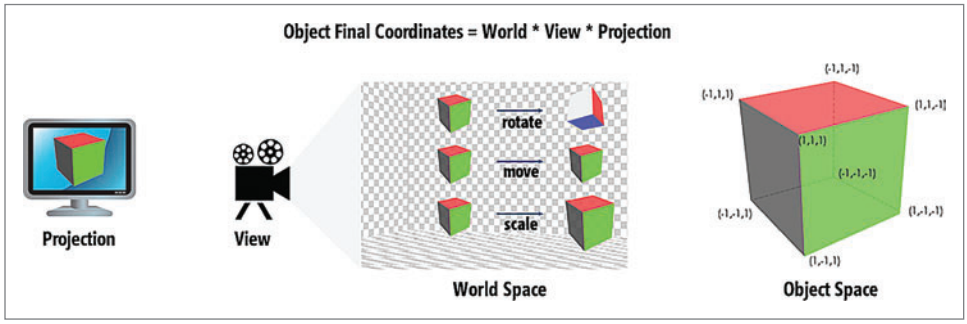


Figure 9 **Coordinates in 3D Space**

The next element is the camera view, which denotes the point from which you're looking at the object. This point can change in the 3D space without changing the coordinates of the actual object in object space as well as world space. To calculate the coordinates of the object with respect to the camera view, you multiply the view matrix with the object's world matrix:

Object View Coordinates =
World Coordinates * View Matrix

Finally, the object view has to be rendered on the screen; this is where you need to calculate the perspective view created because of the distance. So far, my object is in parallel projection (the sides are parallel), but I need to display the object in perspective projection (the sides merge into a vanishing point), so I multiply the product of the object's view matrix and world matrix with the projection matrix:

Object Final Coordinates =
World Coordinates * View Matrix *
Projection Matrix

This is the 3D object's final position on the screen, which is also called the WorldViewProjection.

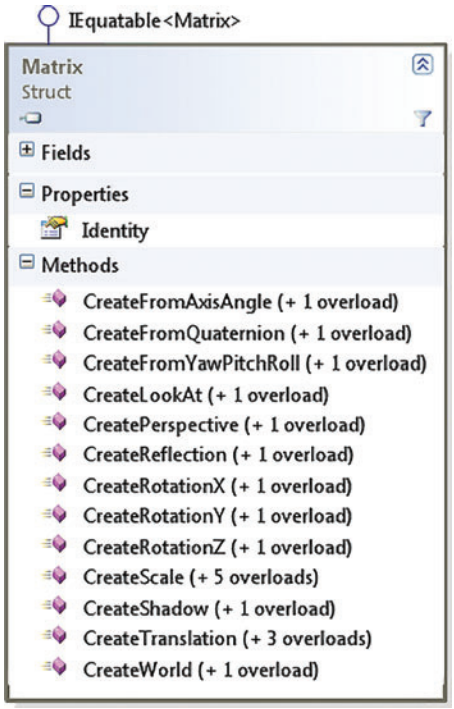


Figure 10 **A Silverlight 5 Matrix Struct**

The Matrix

A Matrix struct, available in Microsoft.Xna.Framework, is included in Silverlight 5. It has a 4x4 homogeneous matrix with 16 floating-point numbers as fields and a number of methods to generate a transformation matrix (see **Figure 10**).

The first three rows by columns (M11- M33) are used for scale and rotate transformations, and the fourth row (M41-M43) is used for translation (see **Figure 11**).

To understand the matrix better, let's see how it's used with respect to a transformation. There are five different types of matrices: 4x4 matrix structure, identity matrix, translation matrix, scale matrix and rotation matrix.

M11	M12	M13	M14
M21	M22	M23	M24
M31	M32	M33	M34
M41	M42	M43	M44

Figure 11 **The 4x4 Matrix**

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

Figure 12 **The Identity Matrix**

Sx	0	0	0
0	Sy	0	0
0	0	Sz	0
0	0	0	1

Figure 13 **The Scale Matrix**

1	0	0	0
0	1	0	0
0	0	1	0
Tx	Ty	Tz	1

Figure 14 **The Translation Matrix**

Identity matrix (see **Figure 12**) is the unit matrix of size 4, and this becomes the original position of the 3D object in world space. If you multiply any matrix with identity matrix, you'll get the original matrix without any change. Matrix structure provides a simple property that returns a Matrix.Identity.

To scale the matrix object, provide a method called Matrix.CreateScale. The scale matrix is used for scale transformation on a 3D object, so when you multiply an object with scale matrix (see **Figure 13**), the resulting matrix resizes accordingly.

The matrix object also provides a Matrix.CreateTranslate method for moving the object in the world space. When multiplied by the translation matrix (see **Figure 14**), the object translates in the world space.

For rotation, there are multiple methods. The Matrix.CreateFromYawPitchRoll method is used for rotating each of the axes for a floating-number value. The Matrix.CreateRotationX, Matrix.CreateRotationY and Matrix.CreateRotationZ methods are for rotating the object along the x,y and z axes. The rotation matrix with respect to angle theta involves M11-M33 elements as shown in **Figure 15**.

Learning the 3D Pipeline for Silverlight-XNA

Silverlight 5 with XNA libraries provides a step-by-step process for creating 3D objects with vertex coordinates for rendering on the screen. This can be divided into five major steps (see **Figure 16**) involving the components shown here:

1. Vertex buffer
2. WorldViewProjection coordinates
3. Shading: vertex, pixel and texture
4. Graphical processing: rasterize, clip and cull
5. Final output: the frame buffer

I'll briefly discuss each step and its components.

1	0	0	0
0	Cos θ	Sin θ	0
0	-Sin θ	Cos θ	0
0	0	0	1

Rotation X

Cos θ	0	Sin θ	0
0	1	0	0
-Sin θ	0	Cos θ	0
0	0	0	1

Rotation Y

Cos θ	Sin θ	0	0
-Sin θ	Cos θ	0	0
0	0	1	0
0	0	0	1

Rotation Z

Figure 15 **Rotation Matrix Along X, Y and Z Axes**

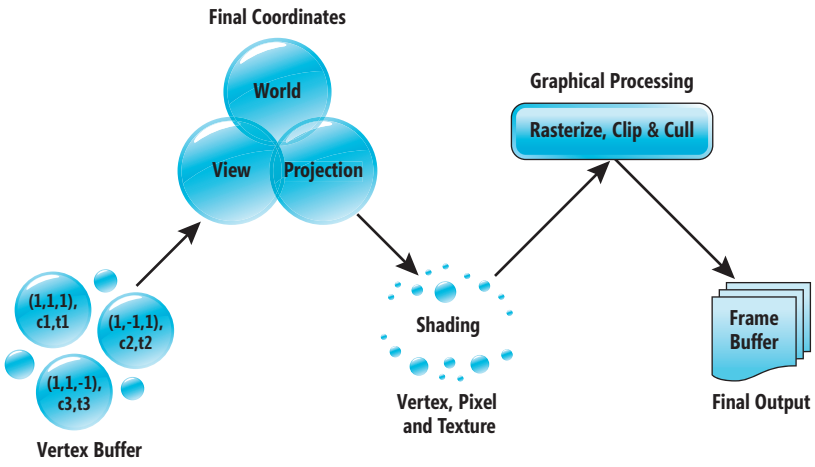


Figure 16 **Creating 3D Objects with the Silverlight 5 XNA Libraries**

Vertex Buffer In creating a vertex buffer from the vertex collection, the first step is to create the skeleton of the 3D object by using a set of vertices. Each vertex contains at least the x, y and z coordinates, but normally also has other properties such as color and texture. This collection of vertices is then used to create a vertex buffer, which goes to the next step in the process.

Silverlight 5 with XNA libraries provides a step-by-step process for creating 3D objects with vertex coordinates for rendering.

WorldViewProjection Coordinates The final coordinates are calculated by multiplying vertices with world, view and projection matrices. Here, the object's relation with the world space, view and projection is calculated and applied. Check the last two sections for more details. Once you have the final coordinates, then the actual shading process takes place.

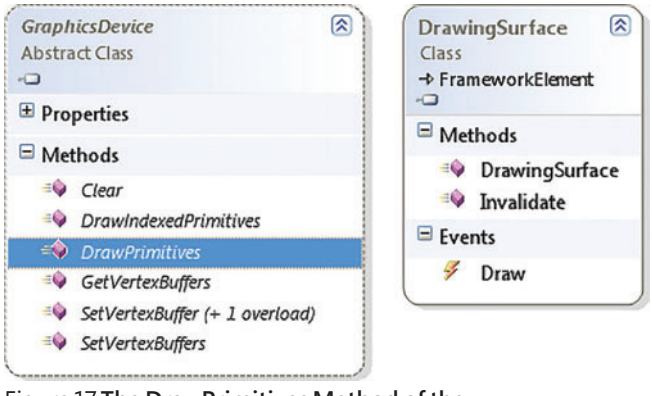


Figure 17 **The DrawPrimitives Method of the GraphicsDevice Class**

Shading Vertex shading, pixel shading and texture shading are used. In this step, first vertex coloring is done, and then the pixel-by-pixel shading happens. Texture shading is also applied in this step. These final shaded coordinates are then used to create a frame buffer.

Rasterize, Clip and Cull During rasterization an image is converted into pixels, and then clip and cull are used to remove the skeleton of the object, along with hidden and invisible layers. This is finally rendered on the screen.

Frame Buffer After the image is rasterized, clipped and culled, a frame buffer is generated, which is then sent to the screen for display.

Creating a Cube Using Primitives

With the knowledge of matrices, world, view, projection and the 3D pipeline in Silverlight 5 with XNA libraries, let's create a 3D cube and see how it all comes together.

The foremost advantage of this approach is the GPU acceleration, which enables hardware acceleration, freeing the CPU from rendering the 3D object. This is enabled by setting the EnableGPUAcceleration parameter in the Object tag to true in the HTML used to configure the Silverlight plug-in, as shown here:

```
<object data="data:application/x-silverlight-2,"
  type="application/x-silverlight-2" width="100%" height="100%">
  <param name="EnableGPUAcceleration" value="true" />
  <param name="source" value="ClientBin/Cube3d.xap"/>
  <param name="minRuntimeVersion" value="5.0.60211.0" />
</object>
```

In XAML, I'll add a DrawingSurface object inside the Grid, which is used to render 3D in Silverlight with the help of the DrawPrimitives method of the GraphicsDevice object (see **Figure 17**):

```
<DrawingSurface Loaded="OnLoad" SizeChanged="OnSizeChanged" Draw="OnDraw"/>
```

Figure 18 **Setting up shaderStream, pixelStream and imageStream**

```
VertexBuffer vertexBuffer;
VertexShader vertexShader;
PixelShader pixelShader;
Texture2D texture;

private void OnLoad(object sender, RoutedEventArgs e)
{
  vertexBuffer = CreateCube();
  Stream shaderStream = Application.GetResourceStream(new
    Uri(@"Cube3d;component/shader/shader.vs", UriKind.Relative)).Stream;
  vertexShader = VertexShader.FromStream(resourceDevice, shaderStream);

  Stream pixelStream = Application.GetResourceStream(new
    Uri(@"Cube3d;component/shader/shader.ps", UriKind.Relative)).Stream;
  pixelShader = PixelShader.FromStream(resourceDevice, pixelStream);

  Stream imageStream = Application.GetResourceStream(new
    Uri(@"Cube3d;component/scene.jpg",
    UriKind.Relative)).Stream;
  var image = new BitmapImage();
  image.SetSource(imageStream);
  texture = new Texture2D(resourceDevice, image.PixelWidth,
    image.PixelHeight, false, SurfaceFormat.Color);
  image.CopyTo(texture);

  Vector3 cameraPosition = new Vector3(0, 0, 5.0f);
  Vector3 cameraTarget = Vector3.Zero;
  view = Matrix.CreateLookAt(cameraPosition, cameraTarget, Vector3.Up);
}
```

Figure 19 **Creating the VertexBuffer in the CreateCube Method**

```
VertexBuffer CreateCube()
{
  var vertexCollection = new VertexPositionColor[36];

  // Front coordinates
  Vector3 cubeA = new Vector3(-0.75f, 0.75f, 0.75f);
  Vector3 cubeB = new Vector3(0.75f, 0.75f, 0.75f);
  Vector3 cubeC = new Vector3(-0.75f, -0.75f, 0.75f);
  Vector3 cubeD = new Vector3(0.75f, -0.75f, 0.75f);
  // Back coordinates
  Vector3 cubeE = new Vector3(-0.75f, 0.75f, -0.75f);
  Vector3 cubeF = new Vector3(0.75f, 0.75f, -0.75f);
  Vector3 cubeG = new Vector3(-0.75f, -0.75f, -0.75f);
  Vector3 cubeH = new Vector3(0.75f, -0.75f, -0.75f);

  // Colors
  Color cRed = Color.FromNonPremultiplied(255, 0, 0, 156);
  Color cGreen = Color.FromNonPremultiplied(0, 255, 0, 156);
  Color cBlue = Color.FromNonPremultiplied(0, 0, 255, 156);
  Color cYellow = Color.FromNonPremultiplied(255, 255, 0, 156);
  Color cBlack = Color.FromNonPremultiplied(0, 0, 0, 156);
  Color cWhite = Color.FromNonPremultiplied(255, 255, 255, 156);

  // Front
  vertexCollection[0] = new VertexPositionColor(cubeA, cGreen);
  vertexCollection[1] = new VertexPositionColor(cubeB, cGreen);
  vertexCollection[2] = new VertexPositionColor(cubeC, cGreen);
  vertexCollection[3] = new VertexPositionColor(cubeB, cBlue);
  vertexCollection[4] = new VertexPositionColor(cubeD, cBlue);
  vertexCollection[5] = new VertexPositionColor(cubeC, cBlue);
  // Back
  vertexCollection[6] = new VertexPositionColor(cubeG, cBlue);
  vertexCollection[7] = new VertexPositionColor(cubeF, cBlue);
  vertexCollection[8] = new VertexPositionColor(cubeE, cBlue);
  vertexCollection[9] = new VertexPositionColor(cubeH, cGreen);
  vertexCollection[10] = new VertexPositionColor(cubeF, cGreen);
  vertexCollection[11] = new VertexPositionColor(cubeG, cGreen);
  // Top
  vertexCollection[12] = new VertexPositionColor(cubeE, cRed);
  vertexCollection[13] = new VertexPositionColor(cubeF, cRed);
  vertexCollection[14] = new VertexPositionColor(cubeA, cRed);
  vertexCollection[15] = new VertexPositionColor(cubeF, cYellow);
  vertexCollection[16] = new VertexPositionColor(cubeB, cYellow);
  vertexCollection[17] = new VertexPositionColor(cubeA, cYellow);
  // Bottom
  vertexCollection[18] = new VertexPositionColor(cubeH, cRed);
  vertexCollection[19] = new VertexPositionColor(cubeG, cRed);
  vertexCollection[20] = new VertexPositionColor(cubeC, cRed);
  vertexCollection[21] = new VertexPositionColor(cubeD, cYellow);
  vertexCollection[22] = new VertexPositionColor(cubeH, cYellow);
  vertexCollection[23] = new VertexPositionColor(cubeC, cYellow);
  // Left
  vertexCollection[24] = new VertexPositionColor(cubeC, cBlack);
  vertexCollection[25] = new VertexPositionColor(cubeG, cBlack);
  vertexCollection[26] = new VertexPositionColor(cubeA, cBlack);
  vertexCollection[27] = new VertexPositionColor(cubeA, cWhite);
  vertexCollection[28] = new VertexPositionColor(cubeG, cWhite);
  vertexCollection[29] = new VertexPositionColor(cubeE, cWhite);
  // Right
  vertexCollection[30] = new VertexPositionColor(cubeH, cWhite);
  vertexCollection[31] = new VertexPositionColor(cubeD, cWhite);
  vertexCollection[32] = new VertexPositionColor(cubeB, cWhite);
  vertexCollection[33] = new VertexPositionColor(cubeH, cBlack);
  vertexCollection[34] = new VertexPositionColor(cubeB, cBlack);
  vertexCollection[35] = new VertexPositionColor(cubeF, cBlack);

  var vb = new VertexBuffer(resourceDevice,
    VertexPositionColor.VertexDeclaration,
    vertexCollection.Length, BufferUsage.WriteOnly);
  vb.SetData(0, vertexCollection, 0, vertexCollection.Length, 0);
  return vb;
}
```

I'll use three methods in the DrawingSurface class to create and render the cube. The OnLoad method is used for creating the cube and initializing all the shaders and the view matrix, which doesn't change in this application. Note that the 3D object is centered at (0,0,0) using 75 percent of the object space with coordinates ranging from (.75,.75,.75) to (-.75,-.75,-.75). Here, I'll create a vertex buffer to hold the vertex collection and initialize the shaderStream, pixelStream and imageStream streams, which will all be used later in the shading step. I'll also initialize the view matrix, which is the angle at which the camera is looking at the object using cameraPosition and cameraTarget with the parameter Vector3.Up (which means the camera is looking upward). This code is shown in **Figure 18**.

Figure 20 **Calling the DrawPrimitives Method to Render the Output Frames**

```
void OnDraw(object sender, DrawEventArgs args)
{
  Matrix position = Matrix.Identity;
  Matrix scale = Matrix.CreateScale(1.0f);
  float xf = 0.0f; float yf = 0.0f; float zf = 0.0f;

  if (cubeXAxis) xf = MathHelper.PiOver4 * (float)args.TotalTime.TotalSeconds;
  if (cubeYAxis) yf = MathHelper.PiOver4 * (float)args.TotalTime.TotalSeconds;
  if (cubeZAxis) zf = MathHelper.PiOver4 * (float)args.TotalTime.TotalSeconds;

  Matrix rotation = Matrix.CreateFromYawPitchRoll(xf, yf, zf);
  Matrix world;
  if (translateZ != 0)
    world = rotation * Matrix.CreateTranslation(0, 0, (float)translateZ);
  else
    world = scale * rotation * position;

  // Calculate the final coordinates to pass to the shader.
  Matrix worldViewProjection = world * view * projection;
  args.GraphicsDevice.Clear(ClearOptions.Target | ClearOptions.DepthBuffer,
    new Microsoft.Xna.Framework.Color(0, 0, 0, 0), 10.0f, 0);

  // Set up vertex pipeline.
  args.GraphicsDevice.SetVertexBuffer(vertexBuffer);
  args.GraphicsDevice.SetVertexShader(vertexShader);
  args.GraphicsDevice.SetVertexShaderConstantFloat4(0, ref worldViewProjection);

  // Set up pixel pipeline.
  args.GraphicsDevice.SetPixelShader(pixelShader);
  args.GraphicsDevice.Textures[0] = texture;
  args.GraphicsDevice.DrawPrimitives(PrimitiveType.TriangleList, 0, 12);
  args.InvalidateSurface();
}
```

The next step is to create the vertex buffer for the 3D cube. I'll create a CreateCube method as shown in **Figure 19** that returns a VertexBuffer. I'll create two rectangles in the 3D space, with ABCD making up the front face of the cube and EFGH the back of the cube. I use the VertexPositionColor structure to create a collection of vertices and the colors associated with each of them.

The OnSizeChanged method of the drawing surface is used to update the projection and the aspect ratio of the screen based on surface dimension:

```
private void OnSizeChanged(object sender, SizeChangedEventArgs e)
{
  DrawingSurface surface = sender as DrawingSurface;
  float sceneAspectRatio = (float)surface.ActualWidth / (float)surface.ActualHeight
  projection = Matrix.CreatePerspectiveFieldOfView
               (MathHelper.PiOver4, sceneAspectRatio, 1.0f, 100.0f);
}
```

The last method is OnDraw, which allows for dynamic transformation on the 3D cube. This is where I apply Matrix.CreateScale to resize the cube, Matrix.CreateFromYawPitchRoll to rotate it and Matrix.CreateTranslate to move it. Here, the worldViewProjection matrix is calculated and sent to the vertexShader method for shad-

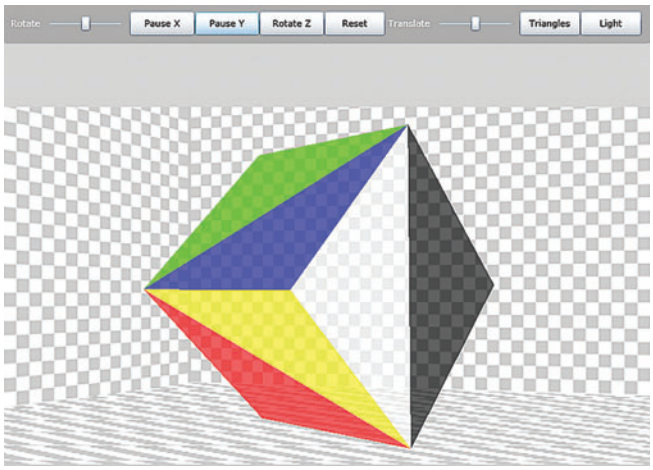


Figure 21 **The Final 3D Cube on the Drawing Surface**

ing the vertices, which in turn transfers to pixelShader to shade the "walls" of the cube. This in turn can also be sent to textureShader, which can be based on an image. Finally, the GraphicDevice Object calls a DrawPrimitives method to render the output frames, as shown in **Figure 20**.

This dynamically renders the final 3D cube on the drawing surface (see **Figure 21**).

> This approach uses GPU acceleration and doesn't render the sides of the 3D object that are hidden or the back side, unlike my first approach.

This approach uses GPU acceleration and doesn't render the sides of the 3D object that are hidden or the back side, unlike my first approach. This also uses the frames buffer in memory to render the final output like the second approach, but with complete control and flexibility of programming against the object, as with the OnDraw method.

You've now learned three different ways to create 3D objects in Silverlight, each with its own strengths and weaknesses. These should be useful additions to your toolbox as you explore the world of 3D in Silverlight. ■

RAJESH LAL *is passionate about Silverlight and Web technologies. He has written multiple books on Windows gadget, Web widget and mobile Web technologies, including an upcoming book, "Fun with Silverlight 4" (CreateSpace, 2011). For more information, contact connectrajesh@hotmail.com or visit silverlightfun.com.*

THANKS to the following technical experts for reviewing this article:
Marc Clifton, Rick Kingslan and Josh Smith

msdn magazine

STATEMENT OF OWNERSHIP, MANAGEMENT AND CIRCULATION

(Required by 39 U.S.C. 3685, United States Postal Service)

1. Title of Publication: MSDN Magazine
2. Publication No. 1528-4859
3. Filing Date: 9/23/11
4. Frequency of Issue: Monthly
5. No. of issues published annually: 12
6. Annual Subscription Price: US $35, International $60
7. Mailing address of known office of publication: 9201 Oakdale Ave., Ste. 101, Chatsworth, CA 91311
8. Mailing address of the headquarters of general business offices of the publisher: Same as above.
9. Name and complete mailing address of Publisher, Editor, and Managing Editor:
   Henry Allain, President, 4 Venture, Ste. 150, Irvine, CA 92618
   Matt N. Morollo, Publisher, 600 Worcester Rd., Ste. 204, Framingham, MA 01702
   Doug Barney, VP/Editorial Director, 600 Worcester Rd., Ste. 204, Framingham, MA 01702
   Wendy Gonchar, Managing Editor, 4 Venture, Ste. 150, Irvine, CA 92618
10. Owner (s): 1105 Media, Inc. dba: 101 Communications LLC, 9201 Oakdale Ave, Ste. 101, Chatsworth, CA 91311. Listing of shareholders in 1105 Media, Inc.
11. Known Bondholders, Mortgagees, and Other Security Holders Owning or Holding 1 Percent or more of the Total Amount of Bonds, Mortgages or Other Securities:
    Nautic Partners V, L.P., 50 Kennedy Plaza, 12th Flr., Providence, RI 02903
    Kennedy Plaza Partners III, LLC, 50 Kennedy Plaza, 12th Flr., Providence, RI 02903
    Alta Communications 1X, L.P., 1X-B, L.P., Assoc., LLC, 28 State St., Ste. 1801, Boston, MA 02109
12. The tax status has not changed during the preceding 12 months.
13. Publication Title: MSDN Magazine
14. Issue date for Circulation Data Below: September 2011
15. Extent & Nature of Circulation:

	Average No. Copies Each Month During Preceding 12 Months	No. Copies of Single Issue Published Nearest to Filing Date
a. Total Number of Copies (Net Press Run)	82,585	82,922
b. Legitimate Paid/and or Requested Distribution		
1. Outside County Paid/Requested Mail Subscriptions Stated on PS Form 3541	71,008	73,144
2. In-County Paid/Requested Mail Subscriptions Stated on PS Form 3541	1,101	1,042
3. Sales Through Dealers and Carriers, Street Vendors, Counter Sales, and Other Paid or Requested Distribution Outside USPS®	7,584	6,383
4. Requested Copies Distributed by Other Mail Classes Through the USPS	0	0
c. Total Paid and/or Requested Circulation	79,693	80,569
d. Nonrequested Distribution		
1. Outside County Nonrequested Copies Stated on PS Form 3541	524	527
2. In-County Nonrequested Copies Distribution Stated on PS Form 3541	0	0
3. Nonrequested Copies Distribution Through the USPS by Other Classes of Mail	0	0
4. Nonrequested Copies Distributed Outside the Mail	2,180	1,644
e. Total Nonrequested Distribution	2,704	2,171
f. Total Distribution	82,397	82,740
g. Copies not Distributed	188	182
h. Total	82,585	82,922
i. Percent paid and/or Requested Circulation	96.72%	97.38%

16. Publication of Statement of Ownership for a Requester Publication is required and will be printed in the November 2011 issue of this publication.
17. I certify that all information furnished on this form is true and complete:
    Abraham Langer, Senior Vice President, Audience Development and Digital Media

JAMES MCCAFFREY

Greedy Algorithms and Maximum Clique

In this month's column, I'll present a solution to the graph maximum clique problem. The solution uses what's called a greedy algorithm, and I'll discuss how to design and test these algorithms. The idea of the maximum clique problem is to find the largest group of nodes in a graph that are all connected to one another. Take a look at the simple graph in **Figure 1**. The graph has nine nodes and 13 edges. The graph is unweighted (there are no priorities associated with the edges) and undirected (all edges are bidirectional). Nodes 2, 4 and 5 form a clique of size three. The maximum clique is the node set 0, 1, 3 and 4, which form a clique of size four.

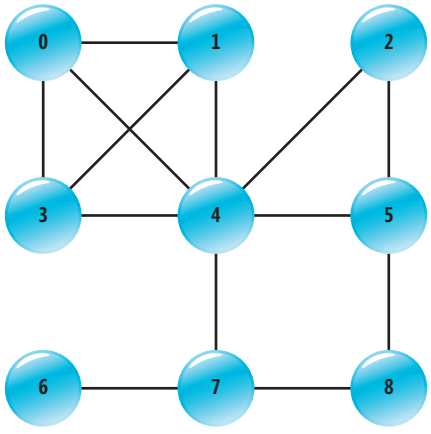


Figure 1 **Graph for a Greedy Maximum Clique Algorithm**

The maximum clique problem is interesting for several reasons. Although it's not apparent from the simple graph in **Figure 1**, the maximum clique problem is one of the most challenging in computer science. It arises in many important applications such as social network communication analysis, where nodes represent people and edges represent a message or relationship. And the maximum clique problem lends itself well to solution by a greedy algorithm, which is a fundamental technique in computer science.

In informal terms, a greedy algorithm is an algorithm that starts with a simple, incomplete solution to a difficult problem and then iteratively looks for the best way to improve the solution. The process is repeated until some stopping condition is reached. **Figure 2** illustrates the important ideas of the maximum clique problem and shows you where I'm headed in this article.

I have a console application, which begins by loading into memory the graph that corresponds to the one shown in **Figure 1**. Designing and coding an efficient graph data structure is a significant problem in itself. I explained the graph data structure code in my last column (msdn.microsoft.com/magazine/hh456397). The algorithm first selects a single node at random, in this case node 2, and initializes a clique with that node. Next, the algorithm looks for the best node to add to the clique. If you refer to **Figure 1**, you'll see that there are only two nodes, 4 and 5, that will form a larger clique. I'll explain what the best node means shortly.

Code download available at code.msdn.microsoft.com/mag201110TestRun.

The algorithm selects and adds node 4 to the clique so the clique is now {2, 4}. At this point, there's only one node in the graph, 5, that will increase the size of the clique, so the algorithm selects node 5 and adds it to the clique. There are now no nodes available that will increase the size of the clique. The algorithm drops the best node from the clique in hopes that a new, different node can be added that will enable progress. Again, I explain what "best" means shortly. The algorithm drops node 4 from the clique. As you can see by looking at the graph, there's no way to make progress. This situation is common with greedy algorithms, so there must be a way to get out of dead-end solutions.

The algorithm is tracking how long it has been since progress has been made—that is, a clique with a larger size has been found. After a certain amount of time with no progress, the algorithm restarts itself. The clique is cleared. This time the algorithm randomly picks node 3 as the initial node. Using the same iterative process to find the best node to add or drop, the algorithm eventually discovers the maximum clique of {0, 1, 3, 4}. In most problems where greedy algorithms are used, the optimal solution isn't known, so the algorithm continues until some stopping condition is met. In this case, the algorithm is configured to stop after 20 iterations. At this point, the best clique found is displayed.

In the sections that follow, I'll walk you through the code that produced the screenshot in **Figure 2**. The complete source code is available at code.msdn.microsoft.com/mag201110TestRun (this column uses the same code from last month). This article assumes you have intermediate-level programming skill with a C-family language or the VB.NET language. I use C#, but I've written the code so that you'll be able to refactor it to another language without too much difficulty if you wish.

Overall Program Structure

The overall structure of the program shown in **Figure 2** is presented in **Figure 3**. The program has dependencies on namespaces System, System.Collections.Generic (a clique is represented as type List<int>), System.IO (for loading the source graph from a text file) and System.Collections (the program-defined MyGraph class uses class BitArray). I rename the main class from the Visual

Studio-template-generated Program to GreedyMaxClique-Program for clarity. I use a class-scope Random object named "random" to initialize and reset a clique to a random node and to break ties when there's more than one best node to add or drop.

The graph is represented as a program-defined MyGraph object. The graph is loaded from an external text file, which uses a standard file format named DIMACS. The key method of MyGraph is AreAdjacent, which returns true if two nodes are connected. I set maxTime to 20 to set an absolute stopping condition for the greedy algorithm. I set targetCliqueSize to establish a second stopping condition; if a clique is found that has the same number of nodes as there are in the graph, the maximum clique must have been found.

The FindMaxClique Method

All of the work is done by the method FindMaxClique, which uses a greedy algorithm to search for the largest clique possible. FindMaxClique calls several helper methods, which I'll describe in detail. I structure the program using static methods, but you may want to refactor the code I present here to a fully object-oriented approach. The FindMaxClique method iterates until one of the two stopping conditions is met and then returns the best clique found. The program definition includes an embedded MyGraph class, which was the subject of last month's article.

In high-level pseudo-code, the FindMaxClique algorithm is:

```
initialize clique with a single node
get list of candidate nodes to add
loop until stop condition met
  if there are nodes that can be added
    find best node to add and add to clique
  else
    find best node to drop and drop from clique

  if lack of progress
    restart algorithm

  update list of candidate nodes
end loop
return largest clique found
```

The FindMaxClique method definition begins:

```
static List<int> FindMaxClique(MyGraph graph, int maxTime,
  int targetCliqueSize)
{
  List<int> clique = new List<int>();
  random = new Random(1);
  int time = 0;
  int timeBestClique = 0;
  int timeRestart = 0;
  int nodeToAdd = -1;
  int nodeToDrop = -1;
...
```

The local clique object is the current clique. I instantiate the Random object using an arbitrary seed value of 1. The time variable is a loop counter variable; because it's discrete, a good alternative name would be "tick." I track the time when the current best clique was found and the time of the last restart to control when restarts will happen. I assign dummy values of -1 to variables for the add-node and drop-node:

```
int randomNode = random.Next(0, graph.NumberNodes);
Console.WriteLine("Adding node " + randomNode);
clique.Add(randomNode);
...
```

The Random.Next(x,y) method returns a value greater than or equal to x and strictly less than y. By setting x = 0 and y = NumberNodes, I get a random node from 0 to 8 inclusive for the example graph:

```
List<int> bestClique = new List<int>();
bestClique.AddRange(clique);
int bestSize = bestClique.Count;
timeBestClique = time;
...
```

The bestClique list is used to track a copy of the largest clique found during the algorithm's search. I use the handy AddRange method to copy the items in the current clique into bestClique. The bestSize variable is used for convenience. The timeBestClique is used to determine if a restart of the algorithm will occur:

```
List<int> possibleAdd = MakePossibleAdd(graph, clique);
List<int> oneMissing = MakeOneMissing(graph, clique);
...
```

The MakePossibleAdd helper method examines the current clique and constructs a list of all nodes that can be added to the clique to increase the clique's size by one. This list is the source of candidates for the best node to add to the clique. The MakeOneMissing helper method is a bit tricky. The method constructs a list of all nodes that are connected to all but one node in the current clique. As you'll see, this oneMissing list is used to determine the best node to drop from a clique. Now I begin the main processing loop:

```
while (time < maxTime && bestSize < targetCliqueSize)
{
  ++time;
  bool cliqueChanged = false;
...
```

As I explained earlier, greedy algorithms typically need one or more stopping conditions. Here the loop terminates if the maximum number of iterations is exceeded or the size of the current

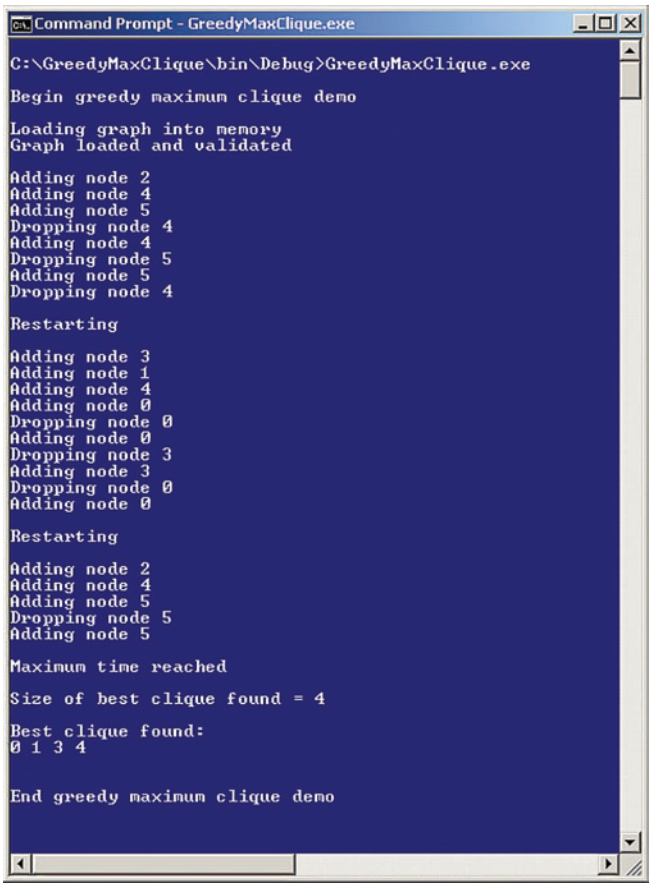


Figure 2 **Greedy Maximum Clique Demo Run**

clique meets a target size. The cliqueChanged variable is used to control the branching logic between adding nodes and dropping nodes. I could've omitted this variable and used an *if..else* control structure instead of separate *if..then* statements, but in this case, the use of a branching control variable leads to code that's easier to read and modify, in my opinion:

```
if (possibleAdd.Count > 0) {
  nodeToAdd = GetNodeToAdd(graph, possibleAdd);
  Console.WriteLine("Adding node " + nodeToAdd);
  clique.Add(nodeToAdd);
  clique.Sort();
  cliqueChanged = true;
  if (clique.Count > bestSize) {
    bestSize = clique.Count;
    bestClique.Clear();
    bestClique.AddRange(clique);
  }
} // add
...
```

I check to make sure there's at least one node that can be added to the clique and then call helper method GetNodeToAdd. This method scans the list of nodes in the possibleAdd list and returns the best node to add (the oft-promised explanation of "best" will be presented when I describe GetNodeToAdd). This node is added to the current clique. At this point I sort the clique, because later in the algorithm I need to search the clique, and if the clique is sorted, I can use a quick binary search instead of a linear search. The new clique is checked to see if it's better (larger) than the current best clique.

Next comes:

```
if (cliqueChanged == false) {
  if (clique.Count > 0) {
    nodeToDrop = GetNodeToDrop(graph, clique, oneMissing);
    Console.WriteLine("Dropping node " + nodeToDrop);
    clique.Remove(nodeToDrop);
    clique.Sort();
    cliqueChanged = true;
  }
} // drop
...
```

If the size of the current clique can't be increased, the algorithm attempts to drop a node from the clique. The helper method GetNodeToDrop selects the best node to drop from the clique:

```
int restart = 2 * bestSize;
if (time - timeBestCliqueFound > restart &&
  time - timeLastRestart > restart) {
   Console.WriteLine("\nRestarting\n");
   timeLastRestart = time;
   int seedNode = random.Next(0, graph.NumberNodes);
   clique.Clear();
   Console.WriteLine("Adding node " + seedNode);
   clique.Add(seedNode);
} // restart
...
```

At this point, the algorithm checks to see if there's been a lack of progress. The restart variable determines how long to wait before restarting. Here I use a value of twice the size of the current best clique. In research on the maximum clique value, the optimal value for restart is still an open question. The value I use here is based on

Figure 3 **Overall Program Structure**

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Collections;

namespace GreedyMaxClique
{
  class GreedyMaxCliqueProgram
  {
    static Random random = null;

    static void Main(string[] args)
    {
      try
      {
        Console.WriteLine("\nBegin greedy maximum clique demo\n");

        string graphFile = "..\\..\\DimacsGraph.clq";
        Console.WriteLine("Loading graph into memory");
        Console.WriteLine("Graph loaded and validated\n");
        MyGraph graph = new MyGraph(graphFile, "DIMACS");

        int maxTime = 20;
        int targetCliqueSize = graph.NumberNodes;

        List<int> maxClique = FindMaxClique(graph, maxTime,
          targetCliqueSize);
        Console.WriteLine("\nMaximum time reached");
        Console.WriteLine("\nSize of best clique found = " +
          maxClique.Count);

        Console.WriteLine("\nBest clique found:");
        Console.WriteLine(ListAsString(maxClique));

        Console.WriteLine("\nEnd greedy maximum clique demo\n");
      }
      catch (Exception ex)
      {
        Console.WriteLine("Fatal: " + ex.Message);
      }
    } // Main

    static List<int> FindMaxClique(MyGraph graph, int maxTime,
      int targetCliqueSize)
    {
      // ...
    }

    static List<int> MakePossibleAdd(MyGraph graph, List<int> clique)
    {
      // ...
    }

    static bool FormsALargerClique(MyGraph graph, List<int> clique,
      int node)
    {
      // ...
    }

    static int GetNodeToAdd(MyGraph graph, List<int> possibleAdd)
    {
      // ...
    }

    static int GetNodeToDrop(MyGraph graph, List<int> clique,
      List<int> oneMissing)
    {
      // ...
    }

    static List<int> MakeOneMissing(MyGraph graph, List<int> clique)
    {
      // ...
    }

    static string ListAsString(List<int> list)
    {
      // ...
    }
  } // class Program

  public class MyGraph
  {
    // ...
  }
} // ns
```

Figure 4 **Make List of oneMissing Nodes**

```
static List<int> MakeOneMissing(MyGraph graph, List<int> clique)
{
  int count;
  List<int> result = new List<int>();
  for (int i = 0; i < graph.NumberNodes; ++i) {
    count = 0;
    if (graph.NumberNeighbors(i) < clique.Count - 1) continue;
    if (clique.BinarySearch(i) >= 0) continue;
    for (int j = 0; j < clique.Count; ++j) {
      if (graph.AreAdjacent(i, clique[j]))
        ++count;
    }
    if (count == clique.Count - 1)
      result.Add(i);
  }
  return result;
}
```

experiments I've performed with several benchmark graph problems. A restart occurs if there's been lack of progress in finding a new best solution or if it's been a relatively long time since the last restart. If a restart is made, the algorithm empties the current clique and then selects a random node from all nodes in the graph. Notice that this is a crude approach; if you refer back to the demo run in **Figure 2**, you'll see that the last restart picked an initial node that had already been used as the first seed node:

```
...
    possibleAdd = MakePossibleAdd(graph, clique);
    oneMissing = MakeOneMissing(graph, clique);
  } // loop
  return bestClique;
}
```

The FindMaxClique method computes from scratch the possibleAdd and oneMissing lists based on the new clique. When the main processing loop terminates, the method returns the best clique found.

Adding the Best Node

There are two steps to determining the best node to add to the current clique. First, a list of all nodes that will increase the size of the current clique is needed. Second, some way to determine which of these nodes is the best node is needed:

```
static List<int> MakePossibleAdd(MyGraph graph, List<int> clique)
{
  List<int> result = new List<int>();
  for (int i = 0; i < graph.NumberNodes; ++i) {
    if (FormsALargerClique(graph, clique, i) == true)
      result.Add(i);
  }
  return result;
}
```

The MakePossibleAdd method scans through each node in the graph. If the node being examined is connected to all nodes in the current clique as determined by helper FormsALargerClique, then the node being examined is a possible "add" node and joins the result list. Notice the result could be an empty list but will never be null:

```
static bool FormsALargerClique(MyGraph graph, List<int> clique, int node)
{
  for (int i = 0; i < clique.Count; ++i) {
    if (graph.AreAdjacent(clique[i], node) == false)
      return false;
  }
  return true;
}
```

The FormsALargerClique method compares one node against all nodes in the current clique. If the candidate node isn't adjacent to even one of the nodes in the clique, then the candidate node can't be added to the current clique. Notice that because AreAdjacent returns false when a node is compared against itself, nodes in the current clique won't be added to the list of possibleAdd nodes.

The main idea behind determining the best node to add is to select the one node from the list of possibleAdd nodes that's most connected to all the other nodes in the possibleAdd set. A node that's highly connected yields the largest possible new list of nodes to add on the next iteration of the algorithm.

Next comes:

```
static int GetNodeToAdd(MyGraph graph, List<int> possibleAdd)
{
  if (possibleAdd.Count == 1)
    return possibleAdd[0];
...
```

The GetNodeToAdd method assumes that the possibleAdd list has at least one node. If there's exactly one node, then that must be the best node:

```
int maxDegree = 0;
for (int i = 0; i < possibleAdd.Count; ++i) {
  int currNode = possibleAdd[i];
  int degreeOfCurrentNode = 0;
  for (int j = 0; j < possibleAdd.Count; ++j) {
    int otherNode = possibleAdd[j];
    if (graph.AreAdjacent(currNode, otherNode) == true)
      ++degreeOfCurrentNode;
  }
  if (degreeOfCurrentNode > maxDegree)
    maxDegree = degreeOfCurrentNode;
}
...
```

There could be several nodes in the possibleAdd list that are tied with others as being the most connected to the other nodes in the list. For example, suppose the graph under analysis is the graph shown in **Figure 1** and the current clique has just node 0. The possibleAdd nodes are {1, 3, 4}. Node 1 is connected to two of the nodes in possibleAdd—and, in fact, so are nodes 3 and 4. So a preliminary scan is made to determine the maximum number of connections available:

```
List<int> candidates = new List<int>();
for (int i = 0; i < possibleAdd.Count; ++i) {
  int currNode = possibleAdd[i];
  int degreeOfCurrentNode = 0;
  for (int j = 0; j < possibleAdd.Count; ++j) {
    int otherNode = possibleAdd[j];
    if (graph.AreAdjacent(currNode, otherNode) == true)
      ++degreeOfCurrentNode;
    }
    if (degreeOfCurrentNode == maxDegree)
      candidates.Add(currNode);
}
...
```

Once the maximum number of connections is known, the algorithm rescans the possibleAdd list and adds each of the nodes in possibleAdd that have the maximum number of connections to a list of candidate nodes. Note that the double-scan could be eliminated by using auxiliary data stores to keep track of the number of connections each possibleAdd node has:

```
...
  return candidates[random.Next(0, candidates.Count)];
}
```

At this point, the candidates list has one or more best nodes to add to the clique. Notice that candidates must have a count of at least one because the possibleAdd list is assumed to have a count of at least one. The algorithm randomly selects one of the candidate nodes and

returns it. I could've put in a check to see if the size of the candidates list is exactly one, and if so, return the single node in candidates.

Dropping the Best Node

Determining the best node to drop from the current clique is a bit subtle. The idea is to drop the one node in the current clique that will result in the largest increase in the list of possibleAdd nodes.

One way to do this is to test each node in the current clique by actually removing it from the current clique and then computing the size of the resulting possibleAdd list. But there's a much more efficient approach that uses a list of nodes that are connected to all but exactly one of the nodes in the current clique. If there is such a oneMissing list of nodes, the list can be used as follows: Scan through each node in the current clique and count how many nodes in the oneMissing list are not connected to the clique node. The node in the current clique that's least-connected to the nodes in the oneMissing list is the best node to drop. After dropping this least-connected node, all the nodes in the oneMissing list that weren't connected to the dropped node will now be fully connected to the remaining nodes in the clique, therefore becoming new possibleAdd nodes.

The MakeOneMissing method is presented in **Figure 4**. The method scans through each node in the graph. The idea is to count how many nodes in the clique are connected to the current node being examined. If the total count of connected nodes is exactly one less than the size of the current clique, then the node being examined is a oneMissing node. The method short-circuits if the current node being examined has fewer than the necessary number of neighbors. The method must filter out nodes that are already in the clique because they're connected to all but one node in their own clique (that is, themselves). Because the current clique is sorted (after each add or drop), a binary search can be used instead of a slower linear search (see **Figure 4**).

The GetNodeToDrop method begins:

```
static int GetNodeToDrop(MyGraph graph, List<int> clique,
 List<int> oneMissing)
{
  if (clique.Count == 1)
    return clique[0];
...
```

The method assumes that the current clique has at least one node. If there's only one node in the current clique, then it's the only node that can be dropped. The method determines the largest number of nodes in the oneMissing list that aren't connected to any node in the current clique because there could be more than one node in the clique that's maximally disconnected to the oneMissing list:

```
int maxCount = 0;
for (int i = 0; i < clique.Count; ++i) {
  int currCliqueNode = clique[i];
  int countNotAdjacent = 0;
  for (int j = 0; j < oneMissing.Count; ++j) {
    int currOneMissingNode = oneMissing[j];
    if (graph.AreAdjacent(currCliqueNode, currOneMissingNode) == false)
      ++countNotAdjacent;
  }
  if (countNotAdjacent > maxCount)
    maxCount = countNotAdjacent;
}
...
```

Once the maximum number of disconnections is known, the method rescans the clique to find those nodes that are most disconnected and adds each to a list of candidates. As with the GetNodeToAdd method, GetNodeToDrop could avoid a rescan by maintaining auxiliary data stores:

```
List<int> candidates = new List<int>();
for (int i = 0; i < clique.Count; ++i) {
  int currCliqueNode = clique[i];
  int countNotAdjacent = 0;
  for (int j = 0; j < oneMissing.Count; ++j) {
    int currOneMissingNode = oneMissing[j];
    if (graph.AreAdjacent(currCliqueNode, currOneMissingNode) == false)
      ++countNotAdjacent;
  }
  if (countNotAdjacent == maxCount)
    candidates.Add(currCliqueNode);
}
...
```

Method GetNodeToDrop finishes by returning a randomly selected node from the list of best nodes to drop:

```
...
  return candidates[random.Next(0, candidates.Count)];
} // GetNodeToDrop
```

Final Thoughts

How effective are greedy algorithms? In spite of their relative simplicity, greedy algorithms have been shown to be quite effective in many problem scenarios. However, effectiveness can be defined using several criteria, including speed and quality of solution. I ran this algorithm against several extremely difficult benchmark graph problems maintained by the DIMACS research organization and the algorithm was quite effective, running quickly and producing maximum cliques that were generally within 5 percent of the best known solutions.

Testing greedy algorithms can be tricky. In addition to all the usual testing techniques such as unit testing, boundary-condition testing and so on—and because greedy algorithms have solutions that grow over time—an effective testing strategy is to write helper methods that iteratively check the state of the data structures used by the algorithm. For example, while testing the maximum clique algorithm presented here, I wrote methods that verify there were no duplicate nodes in the current clique, verify no node in the current clique is in the current possibleAdd or current oneMissing lists and so on.

The greedy maximum clique algorithm I've presented here can be modified to produce better results in several ways. One technique, common to most greedy algorithms, is to add a so-called tabu feature. Tabu algorithms maintain a list of recently used data and possibly a list of recently seen solutions. Data in the tabu list isn't used to construct new solutions. Additionally, greedy algorithms can often employ a strategy called plateau search. When a greedy algorithm is unable to improve its current solution, a plateau search produces a new solution without going backward, such as dropping a node in the maximum clique problem. I'll present these interesting and useful tabu and plateau techniques in a future column. ■

Dr. James McCaffrey *works for Volt Information Sciences Inc., where he manages technical training for software engineers working at the Microsoft Redmond, Wash., campus. He's worked on several Microsoft products, including Internet Explorer and MSN Search. Dr. McCaffrey is the author of ".NET Test Automation Recipes" (Apress, 2006), and can be reached at jammc@microsoft.com.*

Thanks to the following technical experts for reviewing this article: *Paul Koch, Dan Liebling, Ann Loomis Thompson and Shane Williams*

Does your Team do more than just track bugs?

Free Trial and Single User FreePack™ available at www.alexcorp.com

Alexsys Team® does! Alexsys Team 2 is a multi-user Team management system that provides a powerful yet easy way to manage all the members of your team and their tasks - including defect tracking. Use Team right out of the box or tailor it to your needs.

Track all your project tasks in one database so you can work together to get projects done.

- Quality Control / Compliance Tracking
- Project Management
- End User Accessible Service Desk Portal
- Bugs and Features
- Action Items
- Sales and Marketing
- Help Desk

Native Smart Card Login Support including Government and DOD

New in Team 2.11

- Full Windows 7 Support
- Windows Single Sign-on
- System Audit Log
- Trend Analysis
- Alternate Display Fields for Data Normalization
- Lookup Table Filters
- XML Export
- Network Optimized for Enterprise Deployment

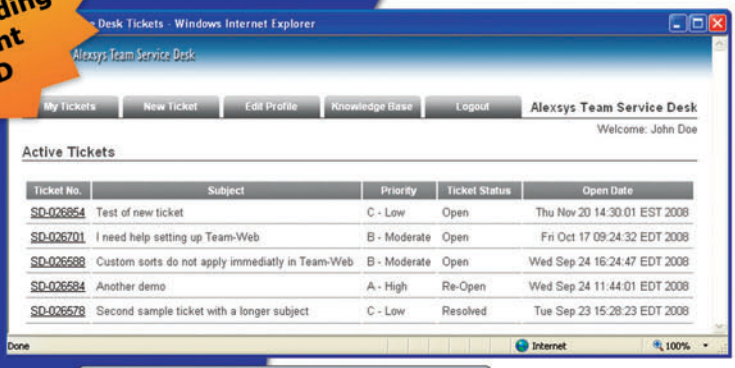
Service Desk Features

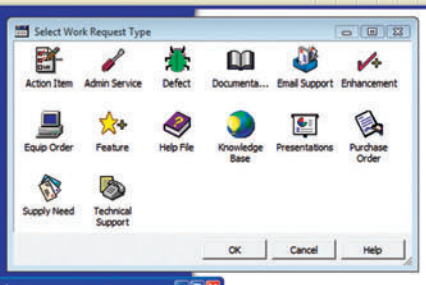
- Fully Secure
- Unlimited Users Self Registered or Active Directory
- Integrated into Your Web Site
- Fast/AJAX Dynamic Content
- Unlimited Service Desks
- Visual Service Desk Builder

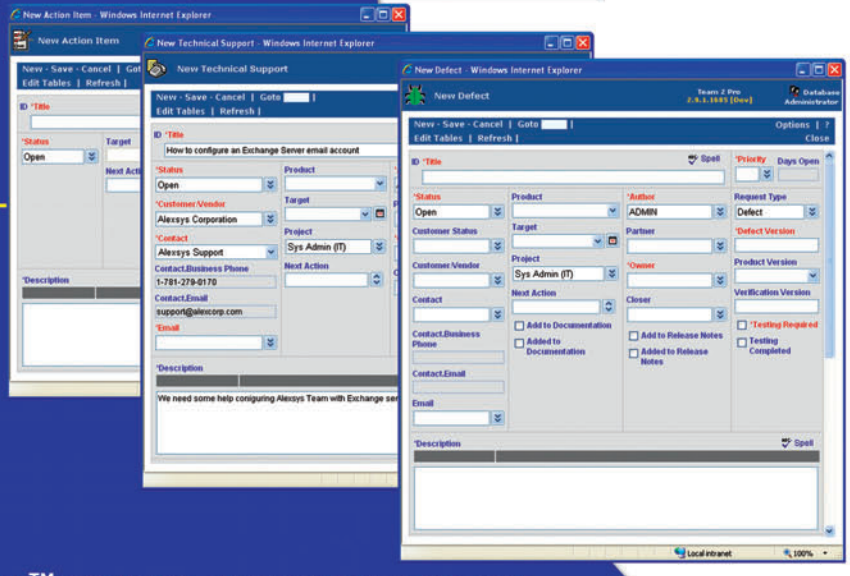
Team 2 Features

- Windows and Web Clients
- Multiple Work Request Forms
- Customizable Database
- Point and Click Workflows
- Role Based Security
- Clear Text Database
- Project Trees
- Time Recording
- Notifications and Escalations
- Outlook Integration

ZDNet Editors' Pick

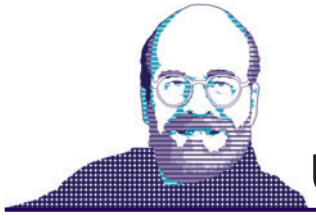






Free Trial and Single User FreePack™ available at www.alexcorp.com.
FreePack™ includes a free single user Team Pro and Team-Web license.
Need more help? Give us a call at 1-888-880-ALEX (2539).

Team 2 works with its own standard database, while Team Pro works with Microsoft SQL, MySQL, and Oracle Servers.
Team 2 works with Windows 7/2008/2003/Vista/XP.
Team-Web works with Internet Explorer, Firefox, Netscape, Safari, and Chrome.



Finishing the E-Book Reader

Before there were Kindles and Nooks and iPads and smartphones; before there was HTML, PDF, XPS, EPUB, MOBI and Plucker; before there were even computers owned by individuals, there was Project Gutenberg. Founded in 1971 by Michael S. Hart (who recently died at the age of 64), Project Gutenberg is easily the oldest collection of digitized public-domain books. While its inventory of about 35,000 books seems quaint when compared to the 15 million or so texts accumulated by Google Books, Project Gutenberg remains an invaluable resource for accessing the classics.

Project Gutenberg books are now available in several rich-text formats, but for many years the focus was entirely on plain text, the reasonable assumption being that rich-text formats come and go but plain text is forever. Five months ago, when I needed a simple but substantial text file for a column demonstrating how to write pagination logic for Windows Phone, I turned to Project Gutenberg.

As I began progressively enhancing this code, the project took on a life of its own. The program has now evolved into a full-fledged e-book reader for Windows Phone that gives you access to the Project Gutenberg library. I call it Phree Book Reader—pronounced "free book reader" and rhyming with "e-book reader" but spelled with a "ph" for "phone"—and it's available as a free download from the Windows Phone marketplace. As usual, you can also download the program's source code from the *MSDN Magazine* Web site.

Catalogs and Web Services

There are many ways to build a large application. Some developers like a top-down approach that begins with the overall structure and gradually implements more detailed code. Others prefer a bottom-up process starting with the low-level routines that are combined into more powerful assemblies.

I tend to mix the two approaches with the primary goal of getting something—anything—working as quickly as possible. Even a skeleton program that's only a tiny bit functional gives me the essential positive feedback that keeps me going, and once I'm using the program, the necessary enhancements become obvious.

If you've been reading this column for the past five months, you know that my previous e-book readers were limited to just one book or, more recently, four books. These books were bundled into the application executable as content. This approach allowed me to focus entirely on the reading experience while avoiding the messy job of downloading books over the Web, and the even messier problem of allowing users to search for books by title and author. But I knew I'd have to meet these challenges eventually.

The Project Gutenberg Web site (gutenberg.org) tries to make it easy for users to search and download books, but it does not implement a public Web service so that other applications can perform similar searches.

Each book stored at Project Gutenberg is identified by an integer ID. If a program knows the ID number for a particular book, it can download an XML file at gutenberg.org/ebooks/N.rdf, where N is the ID number. This resource description format, or RDF, file is often around 10KB in size, and contains the book's title, author and other information, along with links to the book itself in several different formats. This file is great if you know the ID number of a desired book, but not so great otherwise.

Project Gutenberg also makes available a complete catalog of all the books in its collection at gutenberg.org/feeds/catalog.rdf.zip. It's about 9MB in size and unzips to a 200MB XML file. The information in this catalog is similar—but not identical—to the information in the individual RDF files. A new version of the catalog is created every day as new books are added to the Project Gutenberg inventory.

At first I thought my e-book reader could download the entire catalog to the phone and store it in isolated storage for searching purposes, but I was worried about the size. For example, the average word is five characters long and words are separated by spaces, so a 50,000-word book in plain-text format requires only 300KB of storage. The unzipped catalog, in contrast, takes up the same amount of storage as more than 6,000 books!

I then encountered another problem: I could open and read the catalog file with the regular .NET version of XmlReader by setting the ProhibitDtd property of the XmlReaderSettings to false. However, the Silverlight version of XmlReader choked on the file, and no setting of the DtdProcessing property of XmlReaderSettings—or anything else I tried—worked.

After much contemplation, I decided to write my own Web service hosted on my Web site. That Web service obtains the catalog file from the Project Gutenberg site, unzips it, opens it, parses it and then stores a stripped-down version locally in a "flat" format—one text line per book—for faster searching.

Of course, any time you're dealing with somebody else's data, you're also dealing with their data structures. My Web service implements a method named Search with arguments specifying title and author words, and returns up to 25 instances of a type I called

Code download available at code.msdn.microsoft.com/mag201111UIFrontiers.

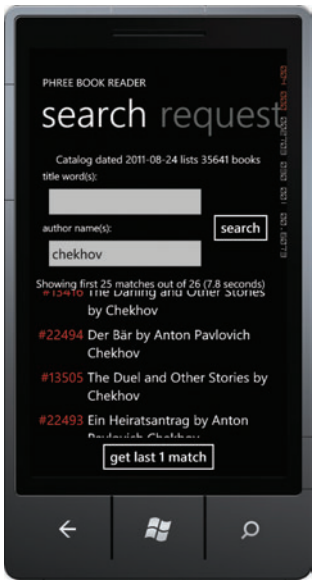


Figure 1 **The Search Item of the Pivot Control**

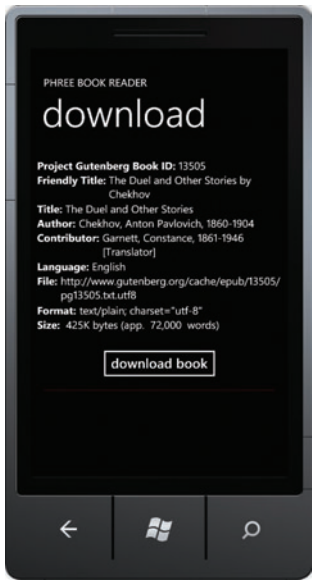


Figure 2 **The Download Page Ready to Download**

GutenbergBook. This class incorporates information about the book obtained from the catalog entry, including a title (and sometimes two titles), zero or more "creators" (author and possibly coauthors), and zero or more contributors (such as translators and editors).

Also included in the Gutenberg catalog is a "friendly title," which generally incorporates both the title and the author, and which is limited to 50 characters. This friendly title seemed ideal for showing search results to the user, and for identifying the book when it's being read.

But this friendly title is not always so friendly. It's great for short titles, such as "Emma by Jane Austen," but is often deficient for longer titles. For example, the Gutenberg catalog contains 12 entries for various editions and volumes of Edward Gibbon's famous historical opus, and all 12 have the same friendly title truncated to 50 characters: "The History of the Decline and Fall of the Roman E."

To me this meant that if I were going to use this friendly title to identify a downloaded book, I'd need to give the user some way to edit it to something more meaningful, such as "Decline and Fall of the Roman Empire, Volume 3."

The Front-End Pivot

At the very least, the front end of an e-book reader needs to display a list of downloaded books and a search screen for downloading more books. It seemed obvious to me that these two items would be part of a Pivot control—a popular control for Windows Phone programs for presenting multiple screens in a format other than navigable pages.

The front-end Pivot control of Phree Book Reader has five items in the following order: bedside, library, search, request and about.

Although search is the third item in the Pivot, it's where a new user will begin. As shown in **Figure 1**, you type in title words or author words, and it makes a call to the Web service. Up to 25 hits are returned and displayed in a ListBox. Each hit is identified by the ID number and friendly title from the Project Gutenberg catalog.

When the user taps one of these items, the program navigates to a download page, as shown in **Figure 2**. This page shows additional information from the catalog and lets you download the book. Notice the Contributor entry indicating the famous translator of Russian literature, Constance Garnett.

When you begin downloading a book, often you'll see the filename suddenly change. The Project Gutenberg catalog contains filenames for the various formats available—including the preferred format for my purposes, plain text encoded in UTF-8—but I discovered that some of these files were empty or corrupted. The filenames in the individual RDF files were different and much more reliable. So before Phree Book Reader begins downloading a book, it downloads the RDF file and obtains a filename from that.

After the book is downloaded, the download page displays buttons that navigate to other pages. The first button lets the user change what the program refers to as the "display title." This title is originally set to the friendly title from the Gutenberg catalog and is also limited to 50 characters.

The second button involves chapter breaks. Project Gutenberg books vary in the number of blank lines used to separate chapters. This option allows a user to change that criterion, and remove superfluous chapter breaks.

The request item on the Pivot control is similar to search except that you simply type in a Project Gutenberg ID number for a book rather than search terms. The Pivot control then navigates to the download page.

When a book has been downloaded, the book joins the library, which is the second item on the Pivot control, and is shown in **Figure 3**.

This library view uses the title and author information from the catalog entry rather than the display title. Books are organized by author and title. Tap one of the titles to start reading that book. Tap the question mark to see the full catalog information (similar to the download page) and optionally edit the display title and the chapter breaks, or delete the book.

There was never any doubt in my mind that I wanted the Phree Book Reader library organized alphabetically by author. That's exactly how my fiction shelves are arranged at home, except I'm not quite so compulsive about alphabetizing the titles. I suppose there might be users of Phree Book Reader who would prefer the library being arranged a little differently, but I wrote this program mostly for myself so I'm really not open to negotiation on this issue!

Displaying books by author and title is a great application of ListBox grouping. The Windows Presentation Foundation (WPF) version of ItemsControl has a terrific property called GroupStyle

Figure 3 **The Library Item of the Pivot Control**

that lets you define a style for a grouping property of items in a CollectionView. In Silverlight, you'd use CollectionViewSource, and although it does support groups, the Silverlight ItemsControl doesn't have a GroupStyle.

Instead, I used ItemsControl for the collection of authors, where the template for each item displays the author's name followed by a non-scrollable ListBox for that author's titles, as shown in **Figure 4**.

Figure 4 **The Library Pivot Item**

```
<ScrollViewer Name="libraryScrollViewer"
              VerticalScrollBarVisibility="Auto">
  <ItemsControl Name="libraryItemsControl">

    <!-- Assumes ItemsSource = AppSettings.Library.Authors -->
    <ItemsControl.ItemTemplate>
      <DataTemplate>
        <StackPanel>
          <!-- Creator -->
          <TextBlock Text="{Binding}"
                     FontWeight="Bold"
                     Margin="0 6" />

          <!-- Books -->
          <ListBox ItemsSource="{Binding Titles}"
                   ItemContainerStyle="{StaticResource listBoxItemStyle}"
                   SelectionChanged="OnLibraryListBoxSelectionChanged">

            <!-- Prevent scrolling of ListBox -->
            <ListBox.Style>
              <Style TargetType="ListBox">
                <Setter Property="ScrollViewer.VerticalScrollBarVisibility"
                        Value="Disabled" />
              </Style>
            </ListBox.Style>

            <ListBox.ItemTemplate>
              <DataTemplate>
                <Grid>
                  <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="Auto" />
                    <ColumnDefinition Width="*" />
                    <ColumnDefinition Width="Auto" />
                  </Grid.ColumnDefinitions>

                  <Grid Grid.Column="0"
                        Margin="12 6"
                        VerticalAlignment="Center">
                    <Polygon Fill="{StaticResource PhoneAccentBrush}"
                             Points="6 0, 47 0, 47 57, 41, 63, 0 63, 0 6" />
                    <Image Source="Images/BookIcon.png" />
                  </Grid>

                  <!-- Book title -->
                  <TextBlock Grid.Column="1"
                             Text="{Binding}"
                             FontSize="{StaticResource
                               PhoneFontSizeMediumLarge}"
                             VerticalAlignment="Center"
                             TextWrapping="Wrap" />

                  <!-- Info Button -->
                  <Button Content=" ? "
                          Grid.Column="2"
                          Tag="{Binding ID}"
                          VerticalAlignment="Center"
                          Click="OnInfoButtonClick" />
                </Grid>
              </DataTemplate>
            </ListBox.ItemTemplate>
          </ListBox>
        </StackPanel>
      </DataTemplate>
    </ItemsControl.ItemTemplate>
  </ItemsControl>
</ScrollViewer>
```

Finding an alternative to this scheme is on my to-do list for Phree Book Reader. I've noticed that as I accumulate more books in the library, the initial load time starts to suffer, and I suspect that the nested ListBox controls are the reason why.

People sometimes refer to books they're currently reading as "the books on my bedside table." For that reason, the first Pivot item is labeled "bedside" and shows a maximum of six books, sorted in descending order of the date and time last read. Each book is identified with its display title.

By the way, a Windows Phone is a great way to read in bed with the lights turned off.

The last Pivot item is labeled "about" and contains some help information about the program as well as links to my past *MSDN Magazine* columns about the e-book reader.

Pivoting Around the Pivots

As I was developing this front end, my biggest struggles involved the Pivot control itself, and navigation away from and back to the Pivot control.

Normally when the program starts up, the default Pivot item should be the bedside. The program should make it as easy as possible to resume reading the most recently read book. However, if the user hasn't read any books yet, the bedside list will be empty so the library view should be first up. And if the user hasn't downloaded any books—perhaps because the program is running for the first time—the search item should be the default.

Programmatically controlling a Pivot control is accomplished by setting the SelectedIndex property of Pivot. However, this property can't be set before the Pivot control has loaded. But after the Pivot control has loaded, setting the property has the effect of visibly sliding the Pivot control to that item. I think I'd prefer a less-visible transition.

Logic involving the Pivot control gets messier when a book is downloaded. If the user navigates from the search item to the download page and then presses the phone's Back key without downloading the book, navigation should go back to the search item. However, if the book is downloaded, then the download page should go back to the library Pivot item where the book is now listed. If the book is downloaded and the user chooses to jump right to the reading view, then navigating back to the home page should cause the bedside item to be visible, again to show the book just read.

I found myself using the State dictionary of the PhoneApplicationService to keep track of where the program has been and what it's doing. The search and request Pivot items set a State dictionary entry named "downloadBook" before navigating to the download page, and that page sets a "successfulDownload" or "successfulDownloadAndRead" dictionary entry depending on whether the user has chosen to jump to the reading view or not. I'm not quite happy with the inelegance of this approach, and perhaps in the future I'll find something that works a little better.

As Usual, Tombstoning

Obviously the larger a program gets, the nastier the tombstoning issues become. Consider the search screen shown in **Figure 1**. I wrote the Web service to return only 25 hits, but also to allow a

program to get an additional 25 with each additional call triggered by the bottom button on the screen. Eventually there could be hundreds—even thousands—of items in the ListBox, depending on the specificity of the search criteria.

This is a good example of a tricky area of tombstoning. All the data in the ListBox could be regenerated by calling the Web service again, but that would take too much time. The items in the ListBox must be saved. But what you do not want to do is save and restore the contents in the OnNavigatedFrom and OnNavigatedTo overrides. This search control is part of a Pivot item on the program's main page, and there's a lot of navigation to and from this page that does not involve tombstoning. It's fine to save and restore small objects in the navigation overrides, but not thousands of items in a ListBox. The ListBox contents should only be saved when the application is actually being tombstoned.

For this program, I experimented with a general-purpose technique for doing precisely this—in the App class I defined a property named TombstoneObjects:

```
public IDictionary<string, object>
TombstoneObjects { set; get; }
```

Any class anywhere in the program can make use of this dictionary. The SearchControl class implements the ITombstonable interface that I discussed in last month's column. In the SaveState method (which is called from the OnNavigationFrom override of MainPage), the control copies the contents of the ListBox to a List object and saves that to the TombstoneObjects dictionary. In the RestoreState method, it restores the contents of the ListBox, but only if the ListBox is empty.

The App class is responsible for saving and restoring the contents of TombstoneObjects to the State dictionary of PhoneApplicationService because the App class has the power to do this intelligently. It knows when the program is being tombstoned because it has installed handlers for the PhoneApplicationService events. The result is that very little extraneous work occurs if the program is not actually being tombstoned.

Although I wrote Phree Book Reader for Windows Phone 7, at the time I'm writing this I've been working with beta versions of the next release. In Windows Phone 7.5, applications are tombstoned less frequently, so it's doubly important for these applications to avoid a lot of unnecessary tombstoning work.

Speaking of the next version of Windows Phone, I've already accumulated a list of features I'd like to add to Phree Book Reader when I subject it to the upgrade. Perhaps we'll be revisiting the program in future columns. ■

CHARLES PETZOLD *is a longtime contributing editor to* MSDN Magazine. *His recent book, "Programming Windows Phone 7" (Microsoft Press, 2010), is available as a free download at bit.ly/cpebookpdf.*

THANKS to the following technical expert for reviewing this article: *Richard Bailey*

DAVID PLATT

BUILD: Microsoft's Call to Arms

Microsoft sure throws a great pep rally. I've just returned from the BUILD conference in Anaheim, Calif., in September, where as everyone knows by now, Microsoft rolled out Windows 8 and its touch-based Metro-style interface. I can't describe all the features in the 700 words I have here, but the buzz, the excitement, and the geeks saying, "Wow, way cool" is something I haven't seen in the Microsoft world for way too long.

I remember attending the Windows 3.1 rollout conference in Seattle in 1991. Microsoft had planned for 500 attendees, but got 2,000. They had to move the sessions from the hotel to a nearby theater, and they needed a parade permit for us to walk to the hotel for meals. That was the first inkling that this Windows thing actually had some legs. Steve Ballmer introduced such groundbreaking technologies as TrueType fonts and object linking and embedding, shouting "Windows, Windows, Windows!" He had more hair then, and so did I.

Microsoft has done some amazing research on the usage patterns of touchscreen users—how they hold a device, where their fingers naturally fall, which motions feel best and worst.

It hasn't been like that lately for Microsoft. The company's stock price stagnated. The U.S. Department of Justice (DOJ) squelched almost anything interesting. Vista flopped. Windows 7 cleaned up the mess, but induced little excitement. Windows Phone 7 earned some praise, but with its tiny market share the platform lacks the cool factor of working on iPhone or Android.

All that has now changed.

Microsoft has done some amazing research on the usage patterns of touchscreen users—how they hold a device, where their fingers naturally fall, which motions feel best and worst. It's carefully constructed a touch language so natural that it's making me annoyed at the Android phone I dearly love. (Press-and-hold for a command menu? Oh, puh-leeze. That's so 2010.) The company is knowing its users, because it knows it's not them.

Windows 8 still faces many challenges. No schedule has been announced, so we know it'll be a while. They're still thinking of tablets as full-fledged PCs, not big phones, which I disagree with. I don't yet see the story for the ultra-light, long battery life, limited function set form factor; which is to say, the fabulously successful iPad market. But Microsoft isn't just copying Apple, imitating. It's thinking things through from first principles, and coming out with really good stuff.

Steve Ballmer made an unannounced appearance at the end of the second keynote. The guy has had a tough decade. Sometimes it seems as if he's aged prematurely, like King Théoden of Rohan in J.R.R. Tolkien's "Lord of the Rings." The U.S. DOJ consent decree must've been a drain, with its constant oversight "whispering subtle poisons" into his ear, much as the evil counselor Wormtongue did to Théoden.

Wormtongue is gone now, the DOJ supervision having expired in May. Here in front of 5,000 amped up geeks, clutching their Windows 8 tablets and hopped up on Jolt Cola, I saw Ballmer feed on their energy. He stood straighter, held his head higher, as his talk built to its climax. "This is the best time ever to be a software developer," he said, and for once in my life, I completely agree with him.

In Tolkien's epic, the wizard Gandalf told Théoden, "Your fingers would remember their old strength better if they grasped a sword hilt," and I saw it happen. Ballmer grasped Microsoft's main weapon, its army of developers—an army of which your humble correspondent has created a tiny part, and helped shape a larger one. The lying video may claim he concluded the speech with his trademark chant of "Developers, developers, developers," but I swear I heard:

Arise now, riders of Théoden!
Dire deeds awake, dark it is eastward
Let horse be bridled, horn be sounded!
Forth Eorlingas!

So now I see a clash of titans: Apple versus Google versus Microsoft. I know whose side I'm on: my own, naturally, same as you are.

I don't flatter myself that Microsoft took the advice in my June column about avoiding DEC's mistakes. (Well, OK, maybe I do.) The company has been working on this strategy for several years. But if anyone wondered whether Microsoft would quietly fade into puzzled obscurity, as DEC did, now you know. It is charging out, swords drawn, shields dressed, lances leveled and helmet visors down.

Win or lose, this battle will be glorious. ■

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.*

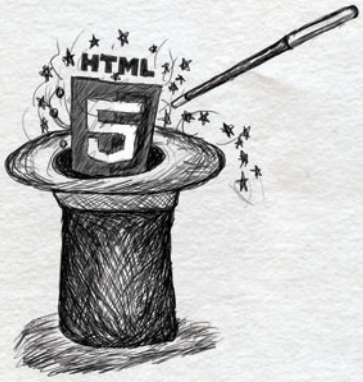

Richard Campbell
Carl Franklin
We Are Smart Developers
.NET Rocks!, the leading Internet audio talk show for .NET developers, recognizes the importance of using the best tools for the job.
GrapeCity PowerTools delivers by providing award-winning Reporting and Spreadsheet components.
Download your FREE trials.
Smarter Components for
SMARTER DEVELOPERS
GrapeCity PowerTools
www.GCPowerTools.com
GvTv.GCPowerTools.com
ActiveReports
Spread.Net
Active Analysis
SD Times 2011 100
Photo Graphic © 2011 F&S Photography

Syncfusion: Essential Studio 2011 Volume 4

Imagine that most of your development work will not be viewed on a desktop; suppose that differences in platforms are now academic. Adopt that view and you'll be ready for what the future holds. We are. That's why Volume 4 provides more cross-platform controls than ever before.

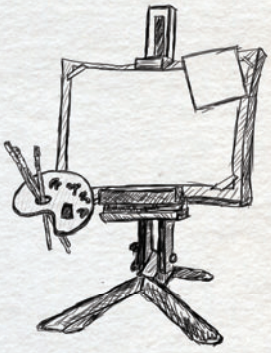
Mobile for the Masses

We're introducing 16 mobile controls that work across Android, iPhone, iPad, and Blackberry devices; our Windows Phone 7 controls add 11 more to the mix. With such an offering, you'll be able to deliver more apps to more people in more locations—embracing the mobile movement.



HTML 5 Magic

The latest incarnation of HTML is set to revolutionize the web experience. For .NET developers, the arena has to be entered early. With over 21 HTML 5-based controls already developed, your entry will be easy, and can be done with the skill set you currently have. When everything is changing, you haven't time to waste.



A New Canvas

Our controls take full advantage of all the richness and interactivity that HTML 5 provides. Pay particular attention to what we've done with our diagram control. The first of its kind for HTML 5, it lends itself fully to the creativity of the user—no plug-in required.

what we mean by 360° transformation

Syncfusion is your development partner—a helping hand outreached. In one ad, we couldn't possibly sell you on all our products; with one magazine, we couldn't possibly convince you of our commitment to service. But we can show you. We invite you to test our full 30-day evaluation—we are confident it is enough. During the trial, please utilize our support to see what we mean by 360° transformation.

Download at **www syncfusion com/downloads/evaluation**

Syncfusion®

Deliver innovation with ease®