

Clarion Magazine

Clarion News

- » Save \$750 on Organize365 Source
- » Clarion Newsgroup Reader Released
- » J-Flow 2.51
- » J-HTML Supports Clarion 5.5
- » CHT Build 12A1.03
- » NetTalk 4.3
- » J-Spell 1.60
- » J-HTML Now A Separate Product
- » StrategyOnline Ides of March Sale
- » Mark Sarson To Run UK CUG
- » Clarion Tech Evangelist Launches ClarionFolk.com
- » SoftDefense Enhances Armadillo
- » vuLimiter 2.0
- » TDC Updated
- » J-Flow 2.41 Adds HTML Controls
- » BST 4.24, BATT 1.0, BBTT 1.1
- » Looking for Gitano Customers
- » Enabling Simplicity Adds Web Hosting
- » Enabling Simplicity Offers Quick Start
- » SetupBuilder 6.7 March 2008 Update
- » AppGen At Aussie DevCon
- » XML Generator blog entry
- » DMC TPS Updating Video
- » DMC 1.1.0.7
- » J-Media Online Tutorial
- » Lodestar 36% Sale in Honor of DST
- » dpQuery 2.10
- » EZRound Tutorial
- » Oak Park Servers Upgraded
- » Wingnut Solutions Acquires Gitano Utilities
- » RightReports 1.06
- » xFiles 1.65
- » FM3 4.30
- » Secwin 4.44
- » Draw 2.61
- » Replicate 2.41
- » Office Inside 2.63
- » New StrategyOnline Web Site
- » DCT2SQL Templates Updated
- » Clarion FreeImage Project Update
- » Clarion Desktop 4.08
- » SetupBuilder Italian Module

[More news]

Save up to **50% off ebooks.**
Subscription has its rewards.



Latest Subscriber Content

Getting The Most Out Of EVALUATE, Part 2

Paul Blais concludes his discussion of EVALUATE with a demonstration of user-defined formulas.

Posted Thursday, March 27, 2008

The Clarion.NET FAQ - Updated March 27, 2008

A list of frequently-asked questions about Clarion.NET/Clarion#, and some hopefully informative answers.

Posted Thursday, March 27, 2008

Getting The Most Out Of EVALUATE

EVALUATE is one of the older functions in the Clarion language. You may know what it does but you probably have not found many times when it works for anything you need to accomplish as a programmer. In this first of two parts Paul Blais explains how EVALUATE works and introduces Evaluator, a tool for testing expressions.

Posted Wednesday, March 26, 2008

Using Interfaces And Composition To Create Flexible Applications

When it comes to designing flexible, adaptable applications, most object-oriented programmers probably think of inheritance and virtual methods first. But interfaces are an equally powerful tool, and when combined with composition you have an excellent toolset for making your applications highly configurable.

Posted Thursday, March 20, 2008

Using Inheritance To Create Flexible Applications

If you've ever had to create two versions of the same application you know the meaning of hair loss. But it doesn't have to be that way; tools and language features are available to ease the development of adaptable, configurable applications, including the mainstay of object-oriented programming, inheritance.

Posted Thursday, March 20, 2008

Writing To The Windows Registry

Richard Rose shows how to update the Windows registry and provides a utility application that lets you change the shortcuts on Windows file dialogs.

Posted Thursday, March 13, 2008

Clarion Magazine Readers In 121 Countries

David Harms provide some information on the distribution of Clarion Magazine readers worldwide, along with the SQL statements he used to extract the data from Clarion Magazine's MySQL database.

Posted Wednesday, March 12, 2008

Source Code Library 2008.02.29 Available

The Clarion Magazine Source Code Library has been updated to include the February source. Source code subscribers can download the Feb 2008 update from the My ClarionMag page. If you're on Vista please run Lindersoft's Clarion detection patch first.

Posted Wednesday, March 05, 2008

[Last 10 articles] [Last 25 articles] [All content]

Source Code

The ClarionMag Source Code Library

Clarion Magazine is more than just a great place to learn about Clarion development techniques, it's also home to a massive collection of Clarion source code. Clarion subscribers already know this, but now we've made it easier for subscribers and non-subscribers alike to find the code they need.

The Clarion Magazine Source Library is a single point download of all article source code, complete with an article cross-reference.

[More info](#) • [Subscribe now](#)

Printed Books & E-Books

E-Books

E-books are another great way to get the information you want from Clarion Magazine. Your time is valuable; with our e-books, you spend less time hunting down the information you need. We're constantly collecting the best Clarion Magazine articles by top developers into themed PDFs, so you'll always have a ready reference for your favorite

- o » [Clarion.NET FAQ](#)
- o » [Clarion# Language Comparison](#)

[\[More Clarion & .NET\]](#)

[\[More Clarion 101\]](#)

Latest Free Content

- o » [The Clarion.NET FAQ - Updated March 27, 2008](#)
- o » [Source Code Library 2008.02.29 Available](#)

[\[More free articles\]](#)

Clarion Sites

- o » [Clarion Jobs](#)

Clarion Blogs

Clarion development topics.

Printed Books

As handy as the Clarion Magazine web site is, sometimes you just want to read articles in print. We've collected some of the best ClarionMag articles into the following print books:



- o » [Clarion Tips & Techniques Volume 4 - ISBN 978-0-9784034-09](#)
- o » [Clarion Tips & Techniques Volume 3 - ISBN: 0-9689553-9-8](#)
- o » [Clarion 6 Tips & Techniques Volume 1 - ISBN: 0-9689553-8-X](#)
- o » [Clarion 5.x Tips and Techniques, Volume 1 - ISBN: 0-9689553-5-5](#)
- o » [Clarion 5.x Tips and Techniques, Volume 2 - ISBN: 0-9689553-6-3](#)
- o » [Clarion Databases & SQL - ISBN: 0-9689553-3-9](#)

We also publish Russ Eggen's widely-acclaimed [Programming Objects in Clarion](#), an introduction to OOP and ABC.

From The Publisher

About Clarion Magazine

Clarion Magazine is your premier source for news about, and in-depth articles on Clarion software development. We publish articles by many of the leading developers in the Clarion community, covering subjects from everyday programming tasks to specialized techniques you won't learn anywhere else. Whether you're just getting started with Clarion, or are a seasoned veteran, Clarion Magazine has the information *you* need.

Subscriptions

While we do publish some free content, most Clarion Magazine articles are for subscribers only. Your [subscription](#) not only gets you premium content in the form of new articles, it also includes all the back issues. Our [search engine](#) lets you do simple or complex searches on both articles and news items. Subscribers can also post questions and comments directly to articles.

Satisfaction Guaranteed

For just pennies per day you can have this wealth of Clarion development information at your fingertips. Your Clarion magazine subscription will more than [pay for itself](#) - you have my personal guarantee.

Dave Harms

ISSN

Clarion Magazine's ISSN

Clarion Magazine's [International Standard Serial Number \(ISSN\)](#) is 1718-9942.

About ISSN

The ISSN is the standardized international code which allows the identification of any serial publication, including electronic serials, independently of its country of publication, of its language or alphabet, of its frequency, medium, etc.

Copyright © 1999-2008 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited.

Clarion Magazine

Clarion News

[Search the news archive](#)

OfficeInside Updated

A new version of OfficeInside is now available. As usual, this is a free upgrade for existing users. Purchase price is \$349. This release includes numerous changes to Excel, Word, Outlook, and overall functionality.

Posted Wednesday, April 02, 2008

Clarion Domains For Sale

Dave Hlavac at Wingnut Solutions has the following Clarion domains for sale: clarionshops.com, clarionutilities.com, and shopclarion.com. Offers entertained.

Posted Wednesday, April 02, 2008

Video Tutorials At StrategyOnline

A number of video tutorials are now available at StrategyOnline. The first three cover J-Html.

Posted Wednesday, April 02, 2008

J-Html 1.03

J-Html 1.03 includes a new control template which adds 31 buttons to the Html Editor.

Posted Wednesday, April 02, 2008

Huenuleufu's Support Pack Includes FileTuner

Huenuleufu's Support Pack now includes FileTuner as a bonus. Save US\$122 with this bundle. Includes: FullRecord, PrintWindow, NeatMessage, WindowID and FileTuner. Source included.

Posted Wednesday, April 02, 2008

PrintWindow 1.20

PrintWindow 1.20 is now available. Changes include: Global and Local options to print current selected tab only (instead of each tab in a new page); Vertical pages moved on PrintWindow Report to avoid controls overlapping when scaling to multiple pages; Option to exclude from printing a range of equate numbers (for excluding run-time generated controls); Color selection print options - normal color, replace colors with Gray or no colors at all; Horizontal page offset to horizontally re-align the content of the page; Embeds prior and after call to print window; Gray and silver background cells in list grids are no longer being printed to avoid Clarion bug with Vista due to number of controls generated; Some manual List boxes were improperly assigned not existing to queues; The "no color" option was not working with some List boxes elements; Better handling of controls out of order (or run-time generated); Vertical lines on Listboxes were improperly drawn when any column had no line; Number of columns on browses were limited by insufficient storage data type; Grids, horizontal lines now not drawn past limit right vertical line.

Posted Wednesday, April 02, 2008

FullRecord 2.09 and 1.87

FullRecord 2.09 is now available. Changes include: Option to workaround Btrieve memo property assignment bug; ABC template fix - "Audit" file name no longer hard coded inside a global procedure; Suppress "previous" error message when inspecting first "change" record without previous read; Fix for some "read" operations, required for changes, not saved on a fresh audit file. FullRecord 1.87 changes include: Option to workaround Btrieve memo property assignment bug; Procedure Name now can be recorded in "source" type procedures; ABC template procedure name generation moved to first priority on Init code section; Suppress "previous" error message when inspecting first "change" record without previous read.

Posted Wednesday, April 02, 2008

Clarion Version Switcher 1.1.8

Clarion Version Switcher version 1.1.8 changes include: Override for Single Instance Lockout (single instance of Clarion); Better support for single instance of CLASwitch; Markers for current use of multiple Clarion versions. You can still only run one instance of each Clarion version but you can open different versions concurrently such as v5.5 and v6. Free download includes C6.1.9033 APP, DCT and icons.

Posted Wednesday, April 02, 2008

Clarion Folk Lore Podcasts

Stu Andrews, official Clarion evangelist, has a couple of podcasts up at the Clarion Folk site.

Posted Wednesday, April 02, 2008

Clarion# Calculator

Owen Brunner has written a calculator program in Clarion#; source is included.

Posted Wednesday, April 02, 2008

gCalc 5.1 Clarion Calculator

Wingnut has released its own branded gCalc utility. Source and icons are included. Changes include: Removed the registration requirement; Installer now installs to the proper 3rdparty directory and registers the template; Improved help file in CHM format; Sample application; Updated template to better support different link methods. This upgrade is a paid upgrade and is available to all users of gCalc, both Gitano and Wingnut customers, no matter the version. Upgrade price is \$49; a new license is \$79. Currently, this is available for C6.0-C6.3 (C55 is pending) including ABC and Legacy support. Demo available.

Posted Wednesday, April 02, 2008

vuSendKeys 1.4 Pre-Release Available

Valutilities is offering a pre-release copy of vuSendKeys 1.4 to all current vuSendKeys registrants. vuSendKeys 1.4 includes seven new functions specifically designed for reading and writing text to a specific control on a specific Window (regardless of what window or control has focus). In order to use these new vuSendKeys functions, you will need to use an external program (such as "Spy++" which ships with Visual Studio or "WinSpy++" a freeware program to glean the ID Number from the Control you want (instructions are included in the vuSendKeys help file).

Posted Wednesday, April 02, 2008

vuLimiter 2.01 Minor Update

An update to vuLimiter (vuLimiter 2.01) is available for download. This update includes an expanded help file, corrects

a spelling error in the default messages, and removes a redundant DLL check that was not necessary.

Posted Wednesday, April 02, 2008

Save \$750 on Organize365 Source

The "Ides of March" sale has been extended to include Organize365 source code. Until next week Friday you can save \$750 off the regular source code price.

Posted Thursday, March 20, 2008

Clarion Newsgroup Reader Released

Organize365 v7.05.2004 is the first version of Organize365 to include newsgroup functionality.

Posted Thursday, March 20, 2008

J-Flow 2.51

J-Flow 2.51 has been released. The HTML controls have been removed (now available as a separate product). As well a compiler warning when using J-Flow and J-Fax together has been fixed.

Posted Thursday, March 20, 2008

J-HTML Supports Clarion 5.5

J-HTML has been successfully tested against Clarion 5.5G. A demo app compiled with Clarion 5.5G is also available.

Posted Thursday, March 20, 2008

CHT Build 12A1.03

CHT Build Update (12A1.03), which is update number three for the first quarter of 2008, has been released.

Posted Thursday, March 20, 2008

NetTalk 4.3

NetTalk 4.30 has been released. This build includes about 35 new features and improvements, and about 30 bug fixes, 6+ new examples, and so on.

Posted Thursday, March 20, 2008

J-Spell 1.60

J-Spell 1.60 is available for download. This release adds the ability to spell-check process procedures.

Posted Thursday, March 20, 2008

J-HTML Now A Separate Product

StrategyOnline has split J-HMTL off as a separate product. During the "Ides of March" sale the discounted price is \$127.50.

Posted Thursday, March 20, 2008

StrategyOnline Ides of March Sale

From March 15-21 StrategyOnline is have an "Ides of March" sale, offering a 15% discount on all its products. All purchases include a three month license for Clarion Desktop. One customer will receive a free license to a Strategy accessory of their choice.

Posted Thursday, March 20, 2008

Mark Sarson To Run UK CUG

Mark Sarson has taken over the running of the UK Clarion User Group from Colin Wynn and is currently liaising with Richard Rose in order to become the owner of the current user group domain. Mark would like to hear from all interested parties with regards to what direction the user group should be going in, and what services it should be offering to its members (How many meetings a year should the group have, when should these be, what subjects would be good for future events etc.). After the successful meeting in Cambridge a couple of years ago, Mark's plan is to expand the user group a little further than its current borders of the UK, so he would also love to hear from anybody within Europe as well. Drop him a line (ukcug (at) marksarson.com) and please supply your full name, email address and contact details.

Posted Thursday, March 20, 2008

Clarion Tech Evangelist Launches ClarionFolk.com

Clarion Folk is the new site from Clarion Tech Evangelist Stu Andrews. This site replaces the Pimp My Clarion site.

Posted Thursday, March 20, 2008

SoftDefense Enhances Armadillo

LANSRAD has released SoftDefense for Clarion developers. SoftDefense is security enhancement for any Clarion developer using the Armadillo/Software Passport software protection system. Using obfuscation techniques and dynamic runtime DLL loading, SoftDefense renders Armadillo-related code virtually invisible to hackers. SoftDefense includes a WEP Key Generator capable of creating 64-bit, 128-bit, 152-bit and 256-bit encryption strings that you can use in your Armadillo Name/Value pairs. SoftDefense works with Clarion 5.5 and 6.x. Since Clarion 7 is an Alpha product support for it is not yet available. The class/templates work with either ABC or Legacy. During the beta status period SoftDefense is \$59.95.

Posted Thursday, March 20, 2008

vuLimiter 2.0

vuLimiter 2.0 has been released, and the download includes a set of demonstration keys so that you can try it before you buy it. The download also includes a licensed test application so that you can test vuLimiter 2.0 for yourself (without having to write your own test app). All current registered users have been sent their update instructions and their individual developer keys. This is a no-cost update for all current registered users of vuLimiter 1.0, and the cost of vuLimiter 2.0 will remain at \$49.00 until the product goes gold (after which the price will increase to \$99.00). This is a major update that includes the ability limit concurrent seats from within the template, or issue a license file that enables additional concurrent seats, enable options, etc.

Posted Friday, March 14, 2008

TDC Updated

A new version of TDC is available for download, valid until June 30. Changes include: Improved speed of access to data; New features; Bug fixes.

Posted Friday, March 14, 2008

J-Flow 2.41 Adds HTML Controls

J-Flow 2.41 includes email controls originally built for Organize365. The HTML control is built using the Microsoft Webbrowser control, and can be used for browsing and editing HTML. You can use the HTML controls without having to use

the J-Flow diagram controls. So J-Flow now consists of an html viewer, and html editor, and a flow diagram control. You can use any one of these on its own, or any combination of them.

Posted Friday, March 14, 2008

BST 4.24, BATT 1.0, BBTT 1.1

BST 4.24 has been released with a year of enhancements and bugfixes. BATT 1.0 has been released with an automatic timer to tame timing activities and provide

Posted Friday, March 14, 2008

Looking for Gitano Customers

Dave Hlavac at Wingnut Solutions Inc., the new owner of gSec, gCalc, gFileFind, gQ, and LGP (Look Good Package) has sent out emails to purchasers of those products. If you are a user of one of the above products and didn't get an email, please contact Dave at feedback at wingnutsolutions.com.

Posted Friday, March 14, 2008

Clarion Magazine

Getting The Most Out Of EVALUATE, Part 2

by Paul Blais

Published 2008-03-27

In [Part 1](#) I wrote static EVALUATE expressions that used runtime properties such as the TODAY() command. Now it's time to go up a level of complexity and code expressions based on user data that is defined at runtime. You can find additional information related to this example in the Eval documentation (PDF or XPS), pages 18 -20.

Substitution within expressions

To accommodate user input I take a base expression and include within it place holders, which I call tokens. At run time I will substitute real data for the tokens.

Evaluator comes with a number of geometric functions; one of the more complex of these computes the area of any triangle. The formula is derived from Heron's formula combined with the Law of Cosines. Evaluator will prompt for the lengths of the three sides and then writes the formula to the Evaluator work space where it can be executed. Here's how you do it.

Step 1: From a blank Evaluator select the Ins button icon (or Alt-I hot key). You'll see the function list shown in Figure 1.

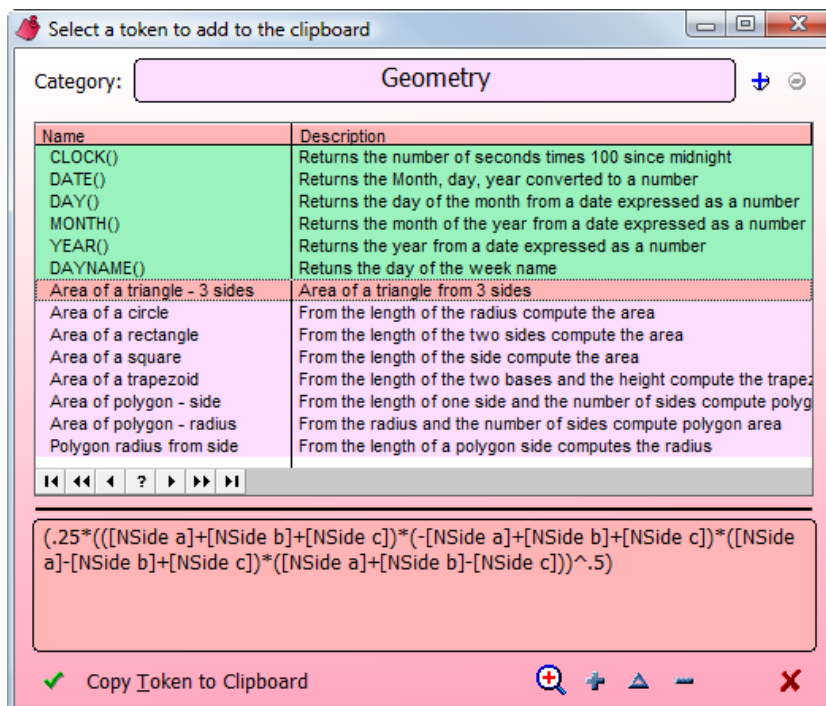


Figure 1. The function list

Step 2: From the list of functions scroll down and select Area of a triangle - 3 sides. Either double-click the formula, or select it and either press Enter or click on the green checkmark

Step 3: Fill in the prompts (Figure 2). For each side a, b, and c I am prompted to enter a length. I'll use the values 3, 4, and 5. EVALUATE does assume valid data, so if I enter something other than numbers it will return an invalid expression error.

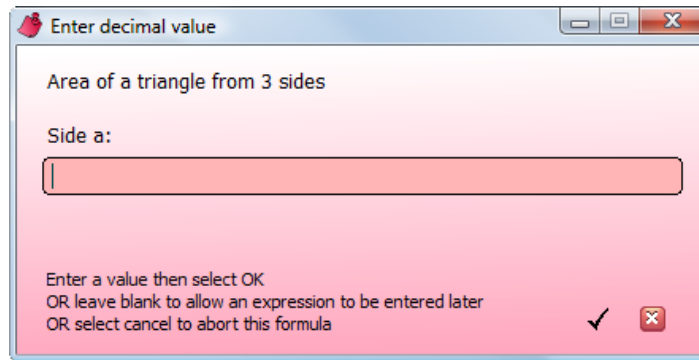


Figure 2. Filling in a prompt

Step 4: Review the expression. After the last prompt the filled-in formula is displayed in the Evaluator workspace. As you can see the values for the three side lengths are each inserted four times. Your typing efficiency is now 4 times greater using the Evaluator!

Step 5: Evaluate the expression. The last step is to press the equal button or the = hot key to display the result. The result is 6 of course. You'll recall that a triangle with sides 3, 4, and 5 units long forms a perfect right triangle, and the area of a right triangle is equal to the two short side multiplied together and divided by 2. So the check for this example is $(3*4)/2$. If I press the CE button (Alt-E) I can enter the check formula as well and get the same result of 6. Using both methods I can now be sure my formula works properly for this test triangle.

Suppose I need to compute the total area of 25 triangular areas so I can estimate the amount of paints I need to buy. Using the above example I type a + in the Evaluator workspace after the formula and insert another triangle formula. I continue adding more triangles to sum all 25 triangles. I also have the benefit of copying the formula and pasting it in a document to save it for later. If I wish I can add comments to the work space by surrounding them with curly braces (as with line breaks, I've written the code to strip out the comments before EVALUATE is called).

Editing formulas

To edit the triangle formula first select it in the function list and then click on the change icon (or press Alt-A). You'll see the window in Figure 3.

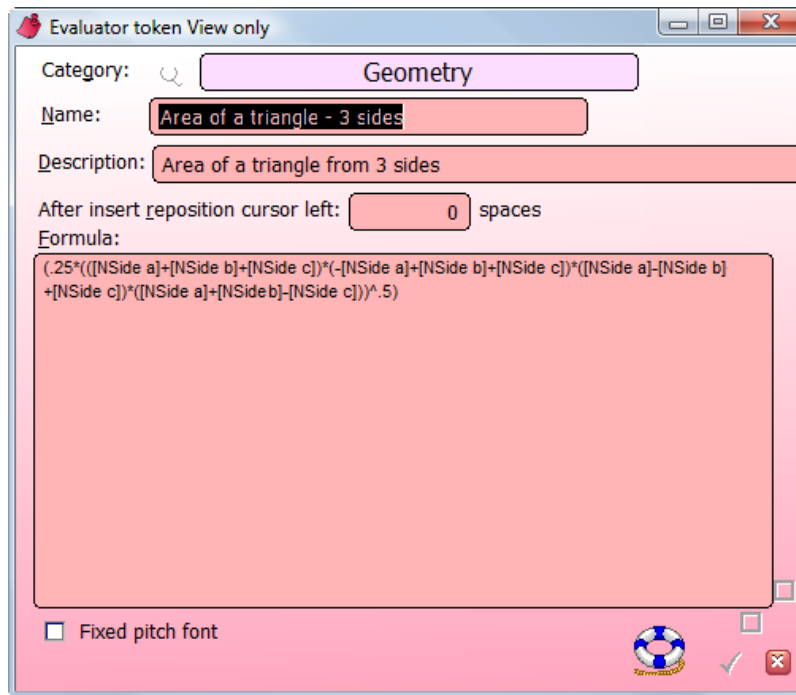


Figure 3. Editing a formula

The category button lets me select a category (or add a new category); the category determines sorting and color coding for the list of functions. The Name field is just a description. The left cursor reposition controls where the cursor ends up after insert, which is useful when I want the cursor to land between two parentheses. After pasting the formula the code moves the cursor.

The formula as displayed may seem confusing at first; here's the generic version where a, b, and c are the lengths of the sides:

$$(.25((a + b + c) * (-a + b + c) * (a - b + c) * (a + b - c)) ^ .5)$$

NOTE: This formula works fine as long as the sum of any two sides is not equal to or greater than to the remaining side. A triangle of sides 1, 2, and 25 won't connect at the three corners.

User defined variables

Evaluator uses some special syntax to create user prompts. As you can see I converted the variables a, b, and c to [NSide A], [NSide B], and [NSide C]. Square brackets cannot really be used in EVALUATE because no array variables exist in its syntax. Evaluator preprocesses the expression and makes a queue of the unique strings surrounded in square brackets. It prompts the user just one time for each unique string, and then substitutes the value entered by the user at every corresponding location in the formula.

The text used for the variables is significant. A first letter of "N" means I want a number; a first letter of I means an integer (see the help PDF for details). The remaining text, "Side A", is the label for the prompt displayed to the user at run time.

Evaluator therefore has some of its own language standards that go beyond what Clarion EVALUATE can do.

Evaluator permits a great many user defined expressions to be created and then later modified or added to.

Where to go from here

EVALUATE is the perfect tool when you need an expression to suit some purpose but you would rather not have it hard coded. It's also a way to allow flexible data output.

I keep the Evaluator program handy for my own calculations as well as testing my Clarion expressions I hard code. It can do anything I could do with a calculator and far more. If I need to write a fancy FORMAT() statement I use real data and try it out in Evaluator first. Coding with expressions is a fundamental idea that gives new features to old applications without a lot of code writing.

[Download Evaluator](#)

[Paul Blais](#) has been a Clarion developer since CPD version 2003. He became a full time independent Clarion Developer in 1998. In 2000 he merged his life and business with his wife Barabra. The merger yielded 3 dogs, one horse, 3 vehicles, and Organizational Development Strategies, Inc.. When not writing code he and his wife can often be seen aboard Bright Eyes sailing the Chesapeake Bay.



Reader Comments

[Add a comment](#)

Clarion Magazine

The Clarion.NET FAQ

by Dave Harms

Published 2007-11-17

In this Frequently Asked Questions (FAQ) page I'll attempt to answer at least some of the questions that have been raised about Clarion.NET and Clarion#. If you log in you can post your own comments and questions below.

Recent additions are **marked in red**; recent deletions are ~~marked with a strikethrough~~.

Terminology!

One of the problems in discussing Clarion.NET is finding a meaningful term for traditional (non-.NET) Clarion applications. Some developers use the term *Win32*, but that isn't always helpful since .NET applications can also be Win32 (or Win64).

~~As noted below, one of the key differences between the Clarion we now know and Clarion.NET is that the former is ultimately built on top of the Windows API. For that reason I will refer to "traditional" Clarion Windows applications as WinAPI apps, to differentiate them from .NET apps.~~

For a while I described Clarion applications as WinAPI apps, but nobody really seemed to like that. In any case the convention now seems to be to describe traditional Clarion apps as simply "Clarion", and Clarion.NET apps as Clarion#, after their respective languages.

~~What did you just say?~~

~~I said that when I use the term *WinAPI app* I'm referring to a traditional Clarion (Windows) application.~~

~~But I don't use the Windows API in my applications.~~

~~You might not, but without the Windows API there would be no Clarion (for Windows) as we know it. All your apps use it all the time.~~

Isn't it too easy to confuse Clarion and Clarion#?

Probably. Oh well.

What's the difference between Clarion# and Clarion.NET?

Clarion# is the language; Clarion.NET is the new IDE. But that's potentially confusing too because the new IDE, when complete, will be used for both Clarion and Clarion# programming. Jan van Dalen suggested the new IDE be called "Clarion Studio" - I like that, and I hope it catches on.

Is Clarion.NET available now?

Yes, the first beta was released to subscription program participants on Saturday, Nov 17, 2007, **and more betas have followed**.

~~What is Clarion.NET?~~

~~Clarion.NET is a version of the Clarion development environment specifically designed for Microsoft's .NET platform.~~

What is .NET, anyway?

From [Wikipedia](#):

The Microsoft .NET Framework is a software component that can be added to or included in the Microsoft Windows operating system. It provides a large body of pre-coded solutions to common program requirements, and manages the execution of programs written specifically for the framework. The .NET Framework is a key Microsoft offering, and is intended to be used by most new applications created for the Windows platform.

I'll just add (for now) that .NET is an object-oriented framework which accommodates a great variety of programming languages.

Isn't .NET all about web development?

You might think so, given the name of the platform. While .NET contains many web/Internet-related classes, it also has extensive support for desktop development. As well, there's a version of the framework for mobile devices.

.NET is a comprehensive development and runtime environment suitable for most kinds of programming, but not usually for low level hardware-oriented stuff (although you *can* get an [80386 assembler](#) that generates .NET code).

What's the difference between Clarion# applications and the Clarion applications I write now?

There are a number of differences. One of the biggest is that the Clarion language (for Windows) is highly dependent on the Windows API. For instance, when you open a window, that window is created by the Clarion runtime library via calls to the Windows API. When you create a new variable or class instance, memory is allocated via the Windows API. This is procedure-oriented coding, and procedure-oriented applications (and operating systems) tend to be hardwired - they have only set ways of doing things. For instance, the Clarion window formatter only has a specific set of controls available, and you can't add your own custom widgets. You can use add-on components like ActiveX controls, but as anyone who's tried it in Clarion can tell you, it isn't all that easy either.

In a Clarion application you often have to work hard to make different functional units of code work together because the Windows API isn't really geared to the concept of components.

Applications written for .NET, however, use the extensive .NET Framework class library rather than the Windows API. In .NET it's all about objects. You can assemble an application out of different classes and components much more easily because .NET provides a congenial framework for all these things to work together.

Is the .NET Framework that much better than the Windows API?

Yes, in almost every way. That's not [toadying](#) to Microsoft - it's just the evolution of programming. The Windows API is disorganized - you need to know what you're looking for or you'll get lost very quickly. The .NET Framework library is organized into namespaces. For instance, security classes are grouped together, as are diagnostic classes, data access classes, etc.

The .NET framework library is much more massive than the API and is growing larger. It provides a whole lot of code that you'd otherwise have to write yourself. There are many other benefits to the library but many of these are better described in the context of the Common Language Runtime (CLR).

Okay, I'll bite. What's the Common Language Runtime?

The CLR is an essential part of .NET. It's the layer that sits between you and the hardware, and which provides important services to your applications including, but not limited to: memory management; thread management; exception handling; garbage collection; security; introspection and reflection. The CLR means many tasks are easier in .NET. For instance, if you NEW something you don't (usually) have to DISPOSE it - the CLR's garbage collector will detect when the object is no longer in use and will clean it up automatically.

Are there any other weird acronyms I should know?

Well, there's CIL, the Common Intermediate Language (often just called IL). The CLR doesn't actually run the code you write; instead, all .NET language compilers translate their code into this CIL (IL) bytecode, and that's what the CLR executes.

So .NET is a giant interpreter? This seems like a step back from true compiled languages - didn't CPD generate pseudocode which had to be interpreted?

You're right - CPD did in fact generate pseudocode, and it's a fair analogy. The advantage of IL is that since all .NET

languages compile to an intermediate language, they all share a common platform. You can take some classes compiled in, say, [Fortran.NET](#) (no, I'm not kidding) and easily use them in C# or [Boo](#) or Clarion#.

What is Clarion#?

Clarion# is the .NET version of the Clarion programming *language*. The Clarion IDE for .NET is, at least at present, called Clarion.NET. But a lot of developers already use the terms Clarion# and Clarion.NET interchangeably.

Clarion is both a procedural and an object-oriented language, but .NET is an OO framework - can I still write procedure code in Clarion#?

Yes, you can still write procedural code in Clarion#. Your procedures are converted to object-oriented code when they're compiled to IL code, but you don't need to know or care about that if procedural code is what makes you happy.

You will, however, find it much more difficult to make the most of the .NET platform and all the available classes unless you are willing to learn at least a little object-oriented programming.

Are Clarion 7 and Clarion# the same product?

No, they are separate products, but they share the same integrated development environment (IDE).

Does that mean if I buy Clarion 7 and Clarion# I'll end up with just one installed IDE able to work with both platforms?

Most likely, although there may be some versioning issues that make that more difficult, at least during the beta process. So for a time you will have two IDEs installed.

What can I do with Clarion.NET/Clarion# beta in the first release?

The beta includes a template wizard you can use to generate a .NET application, but since those templates run in C6, not the new IDE (AppGen isn't ready yet), you can't yet create and maintain a .NET APP file in the same way you do in C6.

As of build 3040, the new IDE ships with a fully functional dictionary editor. The AppGen is scheduled to be demo'd at the Aussie DevCon at the end of May, and Bob Z has indicated AppGen will not be held if it's ready for beta release before then; meanwhile you can hand code not just desktop applications, but also ASP.NET and mobile applications.

There are a number of example solutions shipped with the beta. These include demonstrations of connecting to a database with ADO.NET, displaying a BLOB image onscreen, a mobile app, data binding, drag and drop, a FOREACH with QUEUE example, glass buttons, new listbox features, the SCHOOL application, mixing Clarion# and C#, a simple web service, a "Hello world" ASP.NET web application, a .NET remoting example, the PEOPLE app with data grid and browse procedures, and a screen capture utility.

Since Clarion# is a full .NET producer/consumer language you have full access to all of the .NET framework library as well as the rich supply of third party .NET tools.

What will I be able to do with Clarion.NET/Clarion# in the long run?

When complete Clarion.NET/Clarion# will include templates to generate desktop (WinForms) applications, web (ASP.NET) applications, and mobile (Compact Framework) applications.

What is the Compact Framework?

The Compact Framework is .NET for mobile devices, and includes about 30% of the full framework plus classes specific to mobile devices, and takes up about 1/10 of the space, mostly due to file compression.

What is ASP.NET?

ASP.NET is Microsoft's Active Server Pages web application framework for .NET. You create ASP.NET pages using a development approach similar to desktop development: you place controls on a page, and write code for those controls; ASP.NET then renders the pages accordingly and executes the code on the server side when data comes back from the browser.

How hard is it to write .NET applications?

Writing Clarion# applications can be both easier and harder than writing Clarion apps, and the comparison between the two brings to mind the differences between DOS and Windows API development. You could argue that Windows development was a lot harder because you had to do so much more to create even a simple Hello World application (at least

in C, if not in Clarion). On the other hand, Windows provided standard capabilities like a windowing library; in DOS you had to write or otherwise obtain a windowing library to achieve a consistent, accessible user interface. In DOS you had to know how to talk to every printer you wanted to use; in Windows you talked to the printer driver, and the printer driver sorted out the back end. Similarly .NET does a lot of the heavy lifting you now have to code in Clarion apps, leaving newer and more complex tasks to your wily programming brain.

One big difference between the Windows API and .NET is that the latter is exclusively object-oriented. To get the most out of Clarion# apps you will definitely want some basic OO programming knowledge. With that knowledge in hand, I think you'll find .NET easier and safer to use than the Windows API. If you've had to deal with ActiveX or (heaven help you) directly call COM functions you'll truly find .NET an easier place to code.

.NET introduces some new language concepts that may take some getting used to, such as delegates, which are a sort of type-safe object-oriented callback mechanism.

Is .NET open source?

.NET is not open source, but Microsoft is in the process of making the source code for some parts of the framework public.

Will my third party tools work in .NET?

The majority of third party tools will need to be ported to Clarion#. Templates that don't generate any source code and do not depend on a particular template chain are likely to work without modification, but there aren't many that fall into that category. You're probably best off assuming you'll need new versions until you hear otherwise from the vendor.

I've often said that .NET is a two-edged sword for third party vendors. If what you provide is readily available as part of the .NET framework, then zing!! off goes your head as you step into .NET land. On the other hand, if you have a great product and you can port it to .NET, you're ready to carve a swath through a programming market that numbers in the millions of coders.

Will my customers want .NET and if so why?

Some customers will want .NET just because it's a buzzword. That's an easy sale.

Other customers may or may not know whether they want .NET. Clearly what they want is software that does what they need it to. If their needs run to eye candy or very unique user interfaces, then .NET presents some distinct advantages. There are a bazillion custom controls out there for .NET, all of which you can use with Clarion#. And there are many class libraries that handle important behind-the-scenes tasks as well.

I find it difficult to overstate the importance of ready access to all of that existing code. With Clarion apps you have to be concerned about prototyping functions, register passing conventions, the arcana of COM, etc. etc. With .NET you just drop in the library and start to use it, no matter what language it's written in.

.NET offers programming benefits as well. For instance, your code compiles to IL code which is run under the watchful eye of the CLR. That means the CLR can detect problems with your code and present far more detailed information to you than you get from, say, a GPF in a Clarion application. This makes debugging easier and faster.

Can I run .NET apps on Linux or the Mac? What is Mono?

[Mono](#) is a Novell-sponsored project to port the .NET platform's functionality to multiple platforms, including Linux, Mac OS X, Solaris, BSD, and Windows. Mono necessarily lags behind Microsoft's efforts, and currently has completed support for .NET 1.1 and mostly-complete support for .NET 2.0.

Versions? What are all these .NET versions?

Microsoft released .NET 1.0 in 2002, and 1.1 in 2003. You may see some computers with 1.1 as the latest version, (and some computers without .NET installed at all) but the standard at present is .NET 2.0, released in 2005. Clarion# targets 2.0 apps - there was talk in the early days of 1.1 being supported as well but the only reason I can see for supporting 1.1 is for Mono compatibility, given that Windows Forms 2.0 support is now scheduled for Mono 2.5, which does not have a release target.

For most of us, .NET 2.0 will be the minimum.

What about .NET 3.0? Or 3.5?

The first thing to keep in mind about .NET 3.0 is that it is not a replacement for .NET 2.0. It's a bunch of new stuff added to 2.0, including Windows Presentation Foundation, Windows Workflow Foundation, Windows Communication Foundation, and Windows Card Space. The base class library is unchanged from 2.0.

.NET 3.5 uses the same CLR as 2.0 but it adds some new stuff to the base class library, in particular support for the LINQ query language. 2.0 apps will still run fine on 3.5.

SoftVelocity is "adding support for 3.0/3.5" but no timeline has been indicated.

Will I need to learn C# or VB.NET?

You will not need to learn C# or VB.NET or any other .NET language, but you may want to. In particular there's a lot of C# source code out there, and you may want to adapt some of it to your own uses. If you can read object-oriented Clarion code you won't have much trouble reading C#.

What is ADO.NET?

ADO.NET is Microsoft's data access layer for .NET, and consists of data providers (i.e. drivers) and DataSets. A DataSet is a set of objects that model the database elements (tables, views, columns, rows, relations, etc.).

Will my Clarion (.clw) programs compile in Clarion#?

While much of the Clarion language is unchanged in Clarion#, it's unlikely that any single Clarion application big enough to do useful work will compile as Clarion# code without modification.

What will I have to do to port my apps to .NET?

Porting applications to .NET is a bit of a gray area at the moment. Theoretically it can be done; the question is, is it worth the work? Clarion apps are built on a traditional, client-server model. Is this a good approach to take into the .NET world? Do we really want ABC.NET? Perhaps a multi-tier design would be more appropriate, particularly one where you could easily reuse your business logic in desktop, web, and mobile versions of your application. SV has alluded to this kind of design but it isn't clear yet what kind of desktop application templates will be included with Clarion#.

Can I mix and match .NET objects with Win32 API objects easily?

You can include WinAPI code in a .NET app and vice versa. Easy is a relative term. See Wade Hatler's [series of articles](#).

Can I get my Clarion 7 and earlier programs to use .Net components I create using Clarion# or other .Net languages?

Most likely you could (see the article series above) but I think this would be a stopgap measure at best.

How secure is .NET code? Can it be easily decompiled?

Code security is a legitimate concern in .NET and yes, IL code can be decompiled much more easily than native Windows executable code. .NET obfuscators alter label names and use various code scrambling approaches to make it very difficult for anyone to make sense of the decompiled result. Or you could just hire someone who's a natural at writing unreadable code.

I've heard .NET programs run slower than native code. Will I notice the difference?

.NET uses just-in-time (JIT) compiler technology, meaning that IL code is compiled to native code the first time that IL code is needed by the application. That means there is a small startup penalty but once the code is compiled it runs just like any other native code. Theoretically code produced by a JIT compiler can [outperform](#) code issued by a standard compiler because the JIT compiler can tailor the code to the hardware. *That said, .NET applications often do seem to run slower and take more memory..*

Can I include a .NET runtime with my apps so I don't have to require my customers to install .NET?

There is a tool called the [Salamander .NET Linker, Native Compiler and Mini-Deployment Tool](#) that will do just this. It's a bit expensive, and I don't know how well it works. Apparently [Thinstall](#) will also create self-contained .NET installs.

If both new Clarions deliver desktop applications why should I buy both? Would Clarion# not be enough?

I'll assume here you mean *after* AppGen is released for C7 and Clarion#. While you can create real desktop apps already with Clarion#, most developers will want to use AppGen for larger apps.

So yes, you can create desktop apps with both. Why would you still want C7? Here are a few reasons:

- Better productivity with the new IDE, as compared to C6
- Ability to work with different versions of Clarion within the same IDE
- C7's runtime improvements including eye candy and Unicode/ClearType support.
- Stability - templates are well established and the runtime is solid
- Potentially smaller installs - no need for the .NET runtime

Can a Clarion# class inherit from a C# class?

Definitely. And vice versa. The same goes for all .NET languages.

Does Clarion# support generic types?

Generics are supported, although there are still a few compiler bugs being ironed out as of March 2008.

What are .NET's minimum requirements?

According to [Microsoft](#), the minimum requirements for the .NET 2.0 redistributable are:

- 400 Mhz processor (800 Mhz recommended)
- 96-128 Mb memory (256 or better recommended)
- 280 Mb hard disk space (610 for 64 bit), 1 gig recommended
- 800x600 256 color, 1024x768 high color recommended

As with most Microsoft platform requirements, you're probably not going to be very happy at the low end of the spectrum. I suggest you take the "recommended" values as the minimum values.

Supported x86-based operating systems:

- Microsoft Windows 98
- Microsoft Windows 98 Second Edition
- Microsoft Windows 2000 Professional with SP4
- Microsoft Windows 2000 Server with SP4
- Microsoft Windows 2000 Advanced Server with SP4
- Microsoft Windows 2000 Datacenter Server with SP4
- Microsoft Windows XP Professional with SP2
- Microsoft Windows XP Home Edition with SP2
- Microsoft Windows XP Media Center Edition 2002 with SP2
- Microsoft Windows XP Media Center Edition 2004 with SP2
- Microsoft Windows XP Media Center Edition 2005
- Microsoft Windows XP Tablet PC Edition with SP2
- Microsoft Windows XP Starter Edition
- Microsoft Windows Millennium Edition
- Microsoft Windows Server 2003 Standard Edition
- Microsoft Windows Server 2003 Enterprise Edition
- Microsoft Windows Server 2003 Datacenter Edition Microsoft Windows Server 2003 Web Edition

x64-bit based systems

- Microsoft Windows XP Professional x64 Edition
- Microsoft Windows Server 2003, Standard x64 Edition
- Microsoft Windows Server 2003, Enterprise x64 Edition
- Microsoft Windows Server 2003, Datacenter x64 Edition

Itanium-based systems

- Microsoft Windows Server 2003 with SP1, Enterprise Edition for Itanium-based Systems
- Microsoft Windows Server 2003 with SP1, Datacenter Edition for Itanium-based Systems

Should I be learning C# or VB.NET, or some other .NET language?

Although Clarion is a full-fledged .NET language, it probably will be to your advantage to learn at least one other .NET language. There's a wealth of .NET programming information out there, and the vast majority of books and articles deal with either C# or VB.NET. So which language should you learn?

VB.NET in general has more Clarion-like syntax; there are certainly differences, but VB.NET is more of a "plain English" programming language. C# on the other hand has C-like syntax which isn't to everyone's liking. It's also easier to make non-obvious mistakes with C#. On the other hand, C# is the .NET reference language, so you can expect it to support all the latest .NET features, and it's generally a better source of programming examples.

If you have C, C++, or Java experience, choose C#. If you've never worked in a language with C-like syntax then you'll probably be better off with VB.NET. Carl Barnes recommends [Programming VB .NET: A Guide For Experienced Programmers](#), which is also available as a free download - look for the Free eBook Download link on that page.

Why should I choose Clarion# over Visual Studio and VB.NET or C#?

Clarion developers have enjoyed the benefits of code generation since the days of CPD 2.0. And when the Clarion# AppGen and templates are ready, I think it'll be easy to see the productivity advantage in Clarion#. But the Clarion# AppGen isn't ready yet, and there are no shipping Clarion# templates. Until that happens, why should you choose Clarion# over VB.NET or C#?

First, let me deal with the reasons to use VB.NET or C# instead of, or in addition to, the Clarion# beta. Obviously both those languages are available in gold release, and have been for some years, while Clarion# is, well, in beta. So you can expect fewer bugs in VB.NET and C#, and more complete support for many .NET features. And Visual Studio is a more evolved hand-coder's environment, at least at the moment, with extensive add-in support.

There are, however, some important reasons for choosing the Clarion.NET beta over (or at the very least in addition to) VB.NET and/or C#:

- Language familiarity - you can start getting up to speed on .NET using a language with which you are familiar
- QUEUES - although .NET has extensive support for collections, queues are still a dead-easy way to manage lists in memory, and a terrific feature of the language.
- Reports - The report designer isn't yet feature complete, but the report structure in Clarion# is basically the same as it is in Clarion. Reporting is one of Clarion's great strengths. Creating reports in other .NET languages typically means buying add-on products.
- File access - you have access to all the file drivers, including TopSpeed and Clarion files.
- File processing - you can still use Clarion's file access grammar (SET/NEXT etc.) if you want to.
- String handling - Strings in .NET are [massively different](#) from Strings in Clarion. The ClaString class provides compatibility with Clarion string handling code.
- Preparation - although SV has indicated a C# and/or VB.NET template chain is a future possibility, you can be sure that the first template sets will be for Clarion#. As with Clarion, the better you know the Clarion# language, the better you'll be

able to take advantage of the templates and the corresponding class libraries.

Are there bugs in the beta? Sure, it's a beta. Don't buy in if you're not ready to deal with that fact. But you can already do some pretty cool stuff with the first beta, and the compiler is pretty solid.

Additional reading

- [Clarion Magazine articles on Clarion.NET](#)
- [All Clarion Magazine articles related to .NET](#)
- [Clarion Magazine articles on mixing Clarion 6 and .NET](#)
- [How to subscribe to Clarion Magazine](#)
- [CapeSoft's Clarion.NET FAQ](#)
- [Randy Rogers' Clarion# Examples](#)

David Harms is an independent software developer and the editor and publisher of Clarion Magazine. He is also co-author with Ross Santos of *Developing Clarion for Windows Applications*, published by SAMS (1995), and has written or co-written several Java books. David is a member of the American Society of Journalists and Authors ([ASJA](#)).

Reader Comments

Posted on Saturday, November 17, 2007 by Wolfgang Orth

If both new Clarions deliver desktop applications - why should I buy both? Would Clarion# not be enough? It seems to be the most evolved as it covers Desktop, somehow Web-applications and mobile devices. So the traditional Clarion 7 with its WinAPI-programming is oldfashioned history from now on, even before it got gold? Or am I wrong on this?

.....
Posted on Sunday, November 18, 2007 by Dave Harms

Thanks Wolfgang - see answer above.

Dave

.....
Posted on Wednesday, November 21, 2007 by Carl Barnes

A concern I don't think I see addressed in the FAQ are the system requirements for the End User computer for a Clarion# application.

A "classic" Clarion Win32 app will typically run on Windows 98 or newer. Probably even Windows 95. No patches are needed. Nothing to download. 3rd party tools can change this.

A Clarion# .Net app requires the 2.0 .Net Framework. That is more picky about systems. Here's a list I found:

Windows 2000 SP3; Windows 98; Windows 98 SE; Windows ME; Windows Server 2003; Windows Vista; Windows XP SP2

Note that XP users must have SP2. I don't see NT 4 or 95.

Some (or many) users will have to download and install 2.0. That's a 23MB file that takes a 280MB of disk space. I wonder if the 23MB is just the installer and it downloads more.

In summary if you are trying to reach the maximum users, that may be running older hardware, the .Net way might cost you a few.

.....
Posted on Friday, November 23, 2007 by Dave Harms

Thanks Carl - I've updated the FAQ.

Posted on Thursday, December 13, 2007 by Robert Wright

If one would have to more or less re-write ones apps (and learn clarion#) and it looks like C# and VB.Net are high as recommended languages. What advantage or disadvantage does clarion.net have over visual studio 2005/8? Does any one have a feature comparisson.

Posted on Thursday, December 13, 2007 by Dave Harms

Robert - I've added a response to the FAQ.

Dave

Posted on Wednesday, December 19, 2007 by Robert Wright

How does the tree control in clarion# compare with C6 Clarion, C# and VB.net?

Posted on Wednesday, December 19, 2007 by Dave Harms

Robert,

I've had a look at the declarations in the Clarion.Windows.Forms namespace and don't see anything indicating there's a Clarion-specific tree control. I might have missed it, or something may be in development, or it may be that you'll just use the standard WinForms control or any of the third party products out there.

Dave

Posted on Friday, March 28, 2008 by Stephen Ryan

A clarion specific tree control exists inside the standard clarion list control for dot net.

there is an example in the samples.

[Add a comment](#)

Clarion Magazine

Getting The Most Out Of EVALUATE

by Paul Blais

Published 2008-03-26

EVALUATE is one of the older functions in the Clarion language. You may know what it does but you probably have not found many times when it works for anything you need to accomplish as a programmer.

I look in the toolbox for the tool to get the job done most efficiently and then I move on. EVALUATE does not hit my top three list very often. I think there are two good reasons for this. The first reason is EVALUATE really is a runtime-based tool not a compiler-based tool, as I will demonstrate shortly. The second reason is I write code for the compiler more often than anything else. I tend to think in familiar ways, which can be a barrier to doing things in new and more exciting ways.

How EVALUATE really works

The basic idea of EVALUATE is it accepts a string expression and returns a string result. In this article I'll explore those string expressions, but the critical idea is that EVALUATE does its work at run time and not at compile time. The Clarion runtime library looks over the string input, makes a lot of assumptions, and returns a string containing what it thinks the string expression really means. The words "black box" come to mind here. I don't really know what EVALUATE does or how it does it.

Examine the following code:

```
myVarA    STRING('1')
myVarB    STRING('2')
myResult  STRING(20),DIM(5)
myExpression  STRING(20)

CODE
BIND('myVarA', myVarA)
BIND('myVarB', myVarB)

myResult[1] = EVALUATE('myVarA + myVarB')

myResult[2] = EVALUATE(myVarA & ' + ' & myVarB)

myExpression = myVarA & ' + ' & myVarB
myResult[3] = EVALUATE(myExpression)

myResult[4] = myVarA + myVarB
```

Above I have four different ways to compute myResult. All four ways work and each one returns a string result of '3'. I'll use these four approaches to illustrate some important aspects of EVALUATE. But first a word about BIND.

Using BIND

EVALUATE takes a string expression as input, but on their own string expressions aren't that useful because they don't embody important concepts like variables. To get a variable into a string expression I need to BIND it to a string value. Besides variables I can BIND Clarion operators, Clarion functions or my own custom functions.

The most common syntax for BIND is

```
BIND (name, variable)
```

The following code creates two bound names, myVarA and myVarB, which point to their respective variables.

```
BIND('myVarA', myVarA)
BIND('myVarB', myVarB)
```

The first example makes use of BIND.

Example 1: Using BIND

The first example looks straightforward. The two bound variables are passed to EVALUATE along with the addition operator as a string expression.

```
myResult[1] = EVALUATE('myVarA' + 'myVarB')
```

EVALUATE recognizes the variables as numbers because of BIND and performs the addition.

The bad news about BIND is that it really slows down EVALUATE. The Clarion runtime makes a list of the bound items at compile time and searches the list sequentially at run time to find out if what it sees is in the list; if an item isn't in the list you'll get a "BIND Expression Error". The runtime assumes you forgot to BIND something, while in reality you probably just made a typo. If you make the list long enough you can wait a long time for EVALUATE to finish. Don't BIND when you don't have to! I like to make the Clarion compiler do the work instead of the Clarion runtime whenever I can.

As you will see BIND is not required for the remaining examples, although the next two use it.

Example 2: Operators as strings

The second example uses the original variables, but passes the operator as a string:

```
myResult[2] = EVALUATE(myVarA & '+' & myVarB)
```

One feature of the Clarion language is the ability to automatically cast data types based on the context in which they are used. EVALUATE extends this capability to operators too. By enclosing the addition operator in quotes I trick the compiler into treating the plus sign as raw text and not a mathematical operator. At run time, however, EVALUATE will see a single text string containing not myVarA and myVarB but their contents along with the addition operator:

```
1 + 2
```

My code example creates the variables with static values but at run time it does not matter how the variable values were set.

Example 3: Using an expression variable

The third example is my preferred way and shows the true power of EVALUATE:

```
myExpression = myVarA & '+' & myVarB
myResult[3] = EVALUATE(myExpression)
```

I prefer to write whatever code is required to construct the string expression, then pass that string to EVALUATE. Using

a separate string to contain the expression makes the code easier to read and debug. I also force the compiler to substitute the value at string creation time; in this example I'm passing numbers so don't need to BIND.

Example 4: No EVALUATE needed

I don't use EVALUATE unless I really need to. The fourth example shows an alternative way to add two numbers and return a string result:

```
myResult[4] = myVarA + myVarB
```

Clarion's automatic type conversion takes care of string to integer conversion for the math operator, and also converts the integer result back to a string for storage in myResult[4]. If you rewrite the code above with the variables defined as LONG, REAL, or DECIMAL the code still works exactly the same; the Clarion runtime and the compiler both know how to type cast using the same rules.

It pains me to say that the fourth approach is easier to type and runs faster. EVALUATE isn't very efficient and is hard to use and you type a lot even without BIND. But it's not totally worthless.

With a change of perspective you can see how EVALUATE can do a whole lot more than the above examples.

Using EVALUATE

My above example only does a simple math addition, but EVALUATE can handle all the Clarion math and string functions as well as all the logic and string operators. I cannot use IF, CASE, or EXECUTE structures, but I can use CHOOSE. If I nest CHOOSE statements I can make branch logic embedded in my expressions, and EVALUATE that logic at runtime. These are super expressions since I can make them as long as I like.

Here are my simple rules for using EVALUATE:

1. EVALUATE requires a string input and returns a string - always.
2. Don't BIND when you don't have to.
3. Compiling expressions before passing them to EVALUATE makes for more efficient code.
4. If you know something at compile time you never need to BIND it!
5. Let the Clarion language help you sort out all the type casting to go between strings and other numeric formats.

Demonstrating expression complexity

To help in this demonstration I've included a Clarion Magazine version of Evaluator, a tool I wrote in Clarion 6.3. Evaluator solves literal expressions using EVALUATE. Download the ZIP file from the link at the end of this article, unzip the contents to an empty directory and execute Eval.exe. You will then be ready to follow along.

The quick documentation

Evaluator has a complete set of documentation and you can digest it all at your pleasure. The short training course is quite simple: type expressions in the text box and then click the equal icon or press the equal keyboard key. The result displays on the top result line as you see in Figure 1. The lifesaver icon has PDF and XPS formatted help for the entire program. I will only cover now the things I need you to know to run our example.

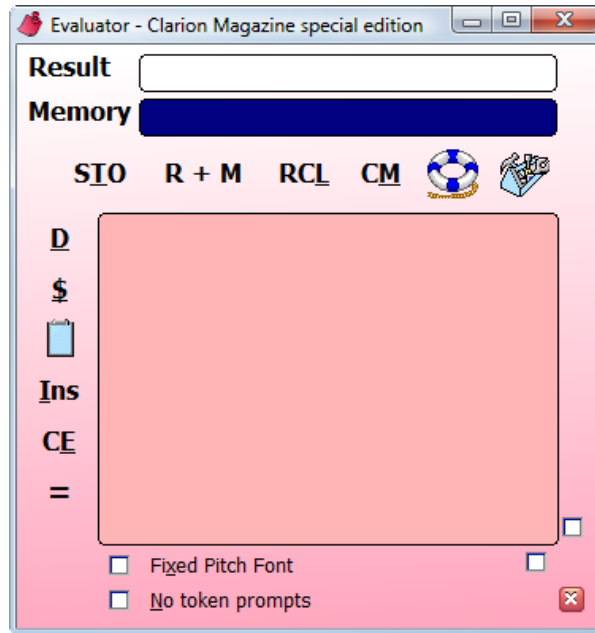


Figure 1. The Evaluator

The scenario

The background for my example is based on a theoretical application that maintains a data record. The record has a description field that is user-defined. It's a sort field but not a primary key field so I allow the user to use anything they want so long as it fits the string length limit. This is the generic case for the use of the application. I want the end user to have the option to provide a default description created from an expression. I will store that expression as a string, call EVALUATE on record insert, and thus prime the record description each time a record is inserted. I want you to assume I could write the record insertion code and I won't cover it here.

Expression concept

That covers the programming needed to add this new default description field. I now want to lay out the concept for the expression I will now write. The mission is to make a default description for a weekly log I will create once per week on Monday of each week reflecting the progress made through the week to the prior day Sunday. I don't want to have to type the description each time, and I want it to follow this format:

Week ending mm/dd/yyyy

Round 1

To write this expression I need to get the date at run time. From the concept above I know that the current date will always be Monday and the desired date is the prior Sunday. Using the Clarion TODAY() function I can subtract one day to know the prior Sunday. In your Evaluator window type the following:

```
'Week ending ' & FORMAT(TODAY()-1,@D2)
```

Note the trailing space after the word 'ending'. Pressing the equal sign will display the result in the evaluator result line. I have met the requirement, except for one problem. Unless you are reading this article on Monday the result is wrong, since the answer displayed is always the date of yesterday relative to now.

My concept was weak. It did not include the idea that I don't always write the log record on Monday because I get lazy. I need a way to make it work when I'm being lazy or so I can run the example on any day of the week and get proper results.

Round 2

I need a way to round the date to the previous Sunday, unless of course if it is Sunday on the day I add the log record. To round the date relative to the day of the week use the modulus (remainder) operator. Type the following into the Evaluator window and press the = sign:

```
TODAY() % 7
```

Sunday modulus 7 is equal to 0, and Monday to Saturday have the values 1 to 6. Here's the expression using the modulus operator to subtract the current day of the week from the value returned by TODAY():

```
'Week ending' & FORMAT(TODAY() - (TODAY()%7) ,@D2)
```

Enter the above expression into your Evaluator window and press equal. The result window now displays the current date rounded to "last Sunday". This must be the correct answer now!

Once again user-collected concept information has been shown to be incorrect. When I really think about how I make these log entries I realize I sometimes make them before the week is done; the above expressions gives incorrect data if I make the log entry early. The rule that my concept should follow is if I make the entry Monday through Wednesday I am creating last week's log, but if I make the log Thursday through Saturday I am making this week's log.

Round 3

The solution is now more complex. There needs to be a logical branch that can round up or down to the coming or prior Sunday. EVALUATE does not know about the Clarion IF statement, but it does know how to resolve CHOOSE.

The new concept will use the same expression as Round 2 except I will add seven days to last Sunday if the current date is greater than Wednesday.

Enter the following expression:

```
'Week ending ' & FORMAT(
TODAY() - TODAY() % 7
+ CHOOSE( TODAY() %7 < 4 , 0, 7)
,@D2)
```

Note that this expression has line feeds. EVALUATE doesn't accept line feed characters, but I've programmed Evaluator to ignore them. Line feeds make it easier for you to test your own expressions in Evaluator.

This expression is identical to Round 2 except for the addition of the CHOOSE clause which adds seven if the modulus 7 of the date is greater than 4. In other words, the solution is computed for last Sunday then rounded forward a week *if* the CHOOSE expression results in a value greater than 4.

Next time

In the [next installment](#) I'll go beyond the basic built-in capabilities of EVALUATE and show how to incorporate user data created at runtime.

[Download Evaluator](#)

Paul Blais has been a Clarion developer since CPD version 2003. He became a full time independent Clarion Developer in 1998. In 2000 he merged his life and business with his wife Barabra. The merger yielded 3 dogs, one horse, 3 vehicles, and Organizational Development Strategies, Inc.. When not writing code he and his wife can often be seen aboard Bright Eyes



sailing the Chesapeake Bay.



Reader Comments

Posted on Friday, March 28, 2008 by Stephen Ryan

nice formulas and a great little demo, i often forget formula's and its nice to see clarion being used for a wide variety of jobs.

thanks for sharing this with us and may the winds be fair for good sailing.

[Add a comment](#)

Clarion Magazine

Using Interfaces And Composition To Create Flexible Applications

by Dave Harms

Published 2008-03-20

In the [first article](#) in this series I explored inheritance as a mechanism for making applications more flexible and adaptable. But there's another technique which is often more useful, and that's to employ a class-like structure called an *interface*, which I'll cover in this article. I'll also discuss the code re-use technique known as composition, and show how you can use both inheritance and interfaces with composition to make applications even more customizable.

Interfaces

Interfaces are really just structures that tell the compiler which methods a class must implement. Think of them as standards, or contracts. They guarantee that a class that implements a given interface can be called in known ways.

In the previous article I showed some sample code that used inheritance to introduce two kinds of functionality, depending on an INI file setting. Here's that code implemented with interfaces:

```
program
```

```
map
```

```
end
```

```
MyInterface    interface
DoSomething    procedure
                end
```

```
ClassA        class,implements(MyInterface),type
                end
```

```
ClassB        class,implements(MyInterface),type
                end
```

```
MyObject    &MyInterface
CustomerNumber long(1)
CustomerA    equate(1)
CustomerB    equate(2)
```

```
code
```

```
!*****!
```

```

! Sample setup.ini contents: !
!           !
! [Customer]           !
! Number=1           !
!*****!
CustomerNumber = GetIni('Customer','Number',CustomerNumber,'./setup.ini')
if CustomerNumber = CustomerA
  MyObject &= new ClassA ! Fails in C6
else
  MyObject &= new ClassB ! Fails in C6
end
MyObject.DoSomething()
dispose(MyObject)

```

```

!*** Class methods ***
ClassA.MyInterface.DoSomething procedure
code
message('ClassA.MyInterface.DoSomething')

ClassB.MyInterface.DoSomething procedure
code
message('ClassB.MyInterface.DoSomething')

```

NOTE: Although I'm showing C6-style code that's compatible with Clarion#, in fact the above code fails under C6 because the compiler can't work out how to assign a class instance to an interface reference. You have to declare a class variable (a &ClassA) and then instantiate the class and assign the interface reference:

```

a &= new ClassA
MyObject &= a.MyInterface

```

Again, that defeats the idea of having a single reference variable for differing class implementations, so clearly Clarion# is a better bet for this kind of work.

Interfaces present a different way to implement varying code for varying needs. You're not deriving (although it's possible to combine inheritance with interfaces); rather you're specifying that your class conforms to a known standard (the INTERFACE) and the calling code can use your class with confidence as it knows the available methods. For a more detailed discussion of the subject see [Understanding OOP Interfaces](#) by David Bayliss.

Composition

In an inheritance-based class design, you plug in your customized code by declaring methods (procedures) for your derived class. Those methods may or may not be virtual methods, depending on whether you need your code to be called by the base class. But while inheritance is vital to any object-oriented language, many applications rely as much or more on something called composition.

Composition is simply one object using a reference to another object. Consider the following example:

```
program
```

```
map
end
```

```
ManagerClass    class,type
MyEmployee      &EmployeeBaseClass,private
AssignEmployee  procedure(EmployeeBaseClass e)
BossEmployeeAround  procedure
                end
```

```
EmployeeBaseClass  class,type
DoSomeWork        procedure,virtual
                end
```

```
EmployeeClassA    class(EmployeeBaseClass),type
DoSomeWork        procedure,virtual
                end
```

```
EmployeeClassB    class(EmployeeBaseClass),type
DoSomeWork        procedure,virtual
                end
```

```
Employee          &EmployeeBaseClass
Manager           &ManagerClass
```

```
EmployeeNumber long(1)
EmployeeA  equate(1)
EmployeeB  equate(2)
```

```
code
```

```
!*****!
```

```
! Sample setup.ini contents: !
```

```
!           !
```

```
! [Employee]           !
```

```
! Number=1           !
```

```
!*****!
```

```
EmployeeNumber = GetIni('Employee','Number',EmployeeNumber,'./setup.ini')
```

```
Manager &= new ManagerClass
```

```
if EmployeeNumber = EmployeeA
```

```
    Employee &= new EmployeeClassA
```

```
else
```

```
    Employee &= new EmployeeClassB
```

```
end
```

```

Manager.AssignEmployee(Employee)
Manager.BossEmployeeAround()
dispose(Employee)
dispose(Manager)

```

```

!*** Class methods ***

```

```

EmployeeBaseClass.DoSomeWork procedure
code
message('EmployeeBaseClass.DoSomeWork')

```

```

EmployeeClassA.DoSomeWork procedure
code
message('EmployeeClassA.DoSomeWork')

```

```

EmployeeClassB.DoSomeWork procedure
code
message('EmployeeClassB.DoSomeWork')

```

```

ManagerClass.AssignEmployee procedure(EmployeeBaseClass e)
code
self.MyEmployee &= e

```

```

ManagerClass.BossEmployeeAround procedure
code
if ~self.MyEmployee &= Null
    self.MyEmployee.DoSomeWork()
else
    message('I can't find my employee!!')
end

```

In this example I have a Manager object which bosses around an employee. Managers need contact with their employees. The declaration for ManagerClass includes this reference variable:

```

MyEmployee &EmployeeBaseClass,private

```

ManagerClass also has a method by which you can assign an EmployeeClass instance (or an instance of a class derived from EmployeeClass) to MyEmployee:

```

ManagerClass.AssignEmployee procedure(EmployeeBaseClass e)
code

```

```
self.MyEmployee &= e
```

In fact this isn't a particular effective manager, as he can only boss around one employee at a time and there are two possible employees. This block of code determines the unlucky employee:

```
if EmployeeNumber = EmployeeA
  Employee &= new EmployeeClassA
else
  Employee &= new EmployeeClassB
end
```

And this code does the actual assignment of the EmployeeClassA or EmployeeClassB instance:

```
Manager.AssignEmployee(Employee)
```

It's time to tell that employee what to do!

```
Manager.BossEmployeeAround()
```

As in the inheritance example, Employee is a reference to the base class type, and EmployeeClassA and EmployeeClassB are derived classes. I could also use interfaces here, subject to the C6 limitations discussed above. I'm only using inheritance here because it's a convenient way to ensure that the Manager knows how to call the DoSomeWork() method. Assuming that employees A and B have completely dissimilar job descriptions, the interface-based approach is cleaner, as I'll show in a moment.

The key point to take away from this example is that I have a generic reference to EmployeeClass inside the Manager class, and I can plug in any matching (in this case derived) class and Manager really won't know the difference - it just calls the method it knows is there, and the employee object does whatever it wants (much like real life).

Composition with interfaces in Clarion#

The following is a Clarion# version of the above example, using interfaces instead of inheritance. There's a little less code involved, and it's clear that this example is all about composition; no inheritance is involved at all. I've removed the unnecessary & characters, but the real reason it will only compile in Clarion# is that Clarion doesn't allow the direct assignment of a class instance to an interface reference.

```
program
```

```
map
end
```

```
ManagerClass    class,type
MyEmployee      EmployeeInterface,private
AssignEmployee  procedure(EmployeeInterface e)
BossEmployeeAround  procedure
                end
```

```
EmployeeInterface  interface
DoSomeWork        procedure
```

```

        end

EmployeeClassA  class,implements(EmployeeInterface),type
        end

EmployeeClassB  class,implements(EmployeeInterface),type
        end

Employee        EmployeeInterface
Manager        ManagerClass

EmployeeNumber long(1)
EmployeeA      equate(1)
EmployeeB      equate(2)

code
!*****!
! Sample setup.ini contents: !
!           !
! [Employee]           !
! Number=1           !
!*****!
EmployeeNumber = GetIni('Employee','Number',EmployeeNumber,'./setup.ini')
Manager = new ManagerClass
if EmployeeNumber = EmployeeA
    Employee = new EmployeeClassA
else
    Employee = new EmployeeClassB
end
Manager.AssignEmployee(Employee)
Manager.BossEmployeeAround()
dispose(Employee)
dispose(Manager)

!*** Class methods ***

EmployeeClassA.EmployeeInterface.DoSomeWork procedure
code
message('EmployeeClassA.EmployeeInterface.DoSomeWork')

EmployeeClassB.EmployeeInterface.DoSomeWork procedure
code
message('EmployeeClassB.EmployeeInterface.DoSomeWork')

```



```

ManagerClass.AssignEmployee      procedure(EmployeeInterface e)
    code
    self.MyEmployee = e

ManagerClass.BossEmployeeAround  procedure
    code
    if ~self.MyEmployee = Null
        self.MyEmployee.DoSomeWork()
    else
        message('I can't find my employee!!')
    end

```

This interface-based implementation doesn't need an EmployeeBaseClass; instead, I create instances of classes that implement EmployeeInterface, and I assign those objects to references that have the EmployeeInterface type.

Inheritance vs composition

When you're designing an application for maximum flexibility, you need to think about the various functional units of code that make up the app and how they can interact. In object-oriented code these units are classes, and as I've hopefully described in this article there are two mechanisms by which you can accomplish this interaction. One is inheritance (often involving virtual methods), and the other is interfaces. Furthermore, objects talk to each other by means of references; if ObjectA has a reference to ObjectB, ObjectA can call that ObjectB's methods. Composition is all about combining otherwise unrelated objects into a larger whole.

In any given application you'll almost always have a mix of inheritance and composition at work. I generally find that my class hierarchies tend to be wide and shallow. If you have a deep hierarchy (class F derived from E derived from D derived from C derived from B derived from A) you'll find that code maintenance becomes more difficult. If you have a bug somewhere in that hierarchy, it's not only harder to track down but it tends to disrupt more code.

Inheritance, particularly in combination with virtual methods, is a great tool, and one I wouldn't want to be without. But I find that I rely more on composition to achieve flexibility and adaptability.

It's a setup

Although inheritance and interfaces make it a lot easier to configure and modify applications on the fly, on their own they still have some limitations. Because you're dealing with data types that must be determined at compile time, the flexibility of your application is somewhat limited by the need to write code to choose between different implementations of classes or interfaces.

Whether you use inheritance, interfaces, or some combination thereof, the critical concept I'm driving toward is the idea of a plug and play architecture that lets you define the behaviour of your application, at least in part, by supplying runtime configuration information rather than hard-coded logic.

I encourage you to start thinking of any one application not as a monolithic block of code, but as a bunch of objects interacting with each other in a variety of ways. This *loose coupling* is part of the larger evolution of programming, and it leads to some important benefits:

- Code reuse: the less any one object is permanently bound to other objects, the easier it is to apply it to new situations.
- Testability: individual test units become smaller and less complex
- Flexibility: loosely coupled objects can more readily be adapted to unexpected situations
- Implementation options: A loosely coupled system can more easily be implemented in a distributed environment.

I've outlined some basic ways you can achieve this kind of architecture in Clarion and Clarion#, but really I'm just setting you up for something much bigger, something that's a vital feature of Clarion# and other .NET languages. That something is *reflection*, and it enables this ideal of loosely coupled objects in some pretty fascinating ways. I'll cover reflection next time.

[David Harms](#) is an independent software developer and the editor and publisher of Clarion Magazine. He is also co-author with Ross Santos of *Developing Clarion for Windows Applications*, published by SAMS (1995), and has written or co-written several Java books. David is a member of the American Society of Journalists and Authors ([ASJA](#)).

Reader Comments

Posted on Friday, March 21, 2008 by Stephen Ryan

I think we found that a class that implements an interface can be used by assigning the new class to the interface.

```
Managerclass.interface &= class2.interface
```

but this does not exit in clarion#

Posted on Monday, March 24, 2008 by Dave Harms

Steve,

Right, in Clarion# you just assign the object to the interface reference.

Thanks for the interface to interface assignment tip.

Dave

[Add a comment](#)

Clarion Magazine

Using Inheritance To Create Flexible Applications

by Dave Harms

Published 2008-03-20

Recently there was a long discussion in comp.lang.clarion ("single useage of a programme"[sic]) which eventually degraded into a variety of discussions including some comments on the nature of .NET programming. One developer opined that the biggest thing to get a handle on is the framework, by which I assume he meant the vast library of classes that make up that framework. In many ways I agree, and I've tried to [make that point](#) in ClarionMag in the past.

But there's also a lot more to .NET than just the class library. There are some important changes in programming styles and techniques, which .NET has appropriated and, in some cases, pioneered.

One of these changes is something called *reflection*. What is it? I'm not going to tell you. Yet. True, it's the greatest thing since sliced bread, programming-wise. But jumping straight into reflection at this point might be too much like Alice walking through the looking glass - the world would probably seem a little bizarre. So rather than take you on that journey right away, I'm going to begin with some things that are *not* reflection, and hopefully by the end of the next article I'll have laid the groundwork for the idea that this something called reflection is a useful solution to the problems I've just described. Along the way I'll provide a brief explanation of inheritance, virtual methods, and interfaces.

Life without reflection

If you've ever developed a piece of software that you sell to multiple customers you've almost certainly encountered the problem of managing competing requirements. In a Clarion application you may be able to accommodate minor changes with a few carefully placed IF or CHOOSE statements; major changes may well require you to create different versions of some procedures. You might use version control software (a good thing) to maintain different versions of your application for different customers, and you might also split up your application into DLLs, to keep code forking to a minimum (another good thing).

Those of us who have been through this experience know that having multiple versions of any block of source code (be it a snippet, procedure, class, or an entire DLL) is a minefield. You'll inevitably find yourself in a situation where you've made a change or fixed a bug in one version of some code but not in another.

There's really just one reason you'll find yourself creating multiple versions of code, and that's because the path of code execution is determined at compile time. Each bit of code in your app is *tightly bound* to other bits of code. That makes sense, right? I mean, you can't just let your application's various pieces of code do whatever they want. When you call a procedure you expect that procedure to be called: when you instantiate a class, you expect just that object and nothing else. Right? Right.

Sort of.

Loosening the ties that bind

The kind of programming Clarion developers do has evolved from procedural pseudocode (CPD) to procedural compiled code

(CDD, Clarion for Windows) to object-oriented code (as of Clarion 4). Even in procedural compiled code it's possible to loosen some of the connections between blocks of code. For instance, a Windows API function that expects a callback function has no idea what procedure you're going to pass; it just assumes you'll give it something that has the right parameters and data types.

For example, if you're using Microsoft's WinInet API to do file uploads you'll want a way to notify the user that the upload is progressing. The following function (taken from Matt Grossmith's [WinInet.DLL: Transferring Files With FTP \(Part 3\)](#)) very simply moves the progress bar to the right each time it is called, and starts over at the beginning once the bar is full::

```

CallbackHandler PROCEDURE(long ConnectionHandle, |
                    long Context, |
                    long Status, |
                    long Info, |
                    long Len)
CODE
! ProgressBarFEQ is the field equate
! of a progress bar on the main
! frame with a range of 0 - 100
ProgressBarFEQ{PROP:progress} = ProgressBarFEQ{PROP:progress} + 1
If ProgressBarFEQ{PROP:progress} >= 100
  ProgressBarFEQ{PROP:progress} = 0
End

```

You then need to tell WinInet DLL to periodically "call back" to your function as the upload progresses. Simply pass the address of your function to the InternetSetStatusCallback API function:

```

AddressOfCallbackHandler = address(CallbackHandler)
ReturnValue = InternetSetStatusCallback( |
  InternetConnectionHandle, |
  AddressOfCallbackHandler)

```

It can be enormously useful to pass your own function to a Windows function, but this technique isn't perfect. You had better get your prototype exactly right; mess up and your app goes Boom! You could use a similar approach to plugging in your own code for custom requirements, but the downside risks are considerable.

Moving along to object oriented code, you can achieve some flexibility in your applications by deciding at runtime which of several classes you'll use for a particular task. One way is to use inheritance.

In an inheritance approach you have a base class which contains the core functionality that you'll always need, and one or more child classes that add some new code. Here's how that works in Clarion and Clarion# (although strictly speaking the Clarion# versions doesn't need the TYPE attributes, nor the & in front of variables, and can use = rather than &= for the reference assignment):

```

program
map
end

```

```

ParentClass    class,type
DoSomething    procedure
                end

```

```

ChildClass     class(ParentClass),type
DoSomethingElse procedure
                end

```

```

p  &ParentClass
c  &ChildClass
CustomerNumber long(1)
CustomerA  equate(1)
CustomerB  equate(2)

```

```
code
```

```
!*****!
```

```
! Sample setup.ini contents: !
```

```
!           !
```

```
! [Customer]           !
```

```
! Number=1           !
```

```
!*****!
```

```
CustomerNumber = GetIni('Customer','Number',CustomerNumber,'./setup.ini')
```

```
if CustomerNumber = CustomerA
```

```
    p &= new ParentClass
```

```
    p.DoSomething()
```

```
    dispose(p)
```

```
else
```

```
    c &= new ChildClass
```

```
    c.DoSomething()
```

```
    c.DoSomethingElse()
```

```
    dispose(c)
```

```
end
```

```
!*** Class methods ***
```

```
ParentClass.DoSomething  procedure
```

```
code
```

```
message('ParentClass.DoSomething')
```

```
ChildClass.DoSomethingElse procedure
```

```
code
```

```
message('ChildClass.DoSomethingElse')
```

In this example I'm checking an INI file for a setting, and depending on that value I'm calling either the base class code or the derived class code. If CustomerNumber has a value of one, I'll see this message:

```
ParentClass.DoSomething
```

If CustomerNumber is not 1, then I'll see these messages:

```
ParentClass.DoSomething
```

```
ChildClass.DoSomethingElse
```

The first method is actually the parent class method, and the second is the child class method.

This isn't, however, a very good example, because although I reuse the method from the original code, I also have to explicitly call the extra method, which means that I have two points of maintenance: the class method, and the code that calls the class method. Wouldn't it be nice if I could just plug in my new class somehow and have the additional code called automatically?

Yes, it would. And the ability to do that is provided by something called a virtual method.

Virtual methods

Here's the previous example slightly modified:

```
program
```

```
map
```

```
end
```

```
ParentClass    class,type
```

```
DoSomething    procedure
```

```
DoSomethingElse procedure,virtual
```

```
end
```

```
ChildClass    class(ParentClass),type
```

```
DoSomethingElse procedure,virtual
```

```
end
```

```

MyObject  &ParentClass
CustomerNumber long(1)
CustomerA  equate(1)
CustomerB  equate(2)

code
!*****!
! Sample setup.ini contents: !
!           !
! [Customer]           !
! Number=1           !
!*****!
CustomerNumber = GetIni('Customer','Number',CustomerNumber,'./setup.ini')
if CustomerNumber = CustomerA
    MyObject &= new ParentClass
else
    MyObject &= new ChildClass
end
MyObject.DoSomething()
dispose(MyObject)

!*** Class methods ***
ParentClass.DoSomething  procedure
code
message('ParentClass.DoSomething')
self.DoSomethingElse

ParentClass.DoSomethingElse procedure
code
message('ParentClass.DoSomethingElse')

ChildClass.DoSomethingElse procedure
code
message('ChildClass.DoSomethingElse')

```

Note that I no longer have two instance variables: I just have one, called MyObject. Depending on the value of CustomerNumber I either create MyObject as an instance of ParentClass or ChildClass. I can always assign a derived class to

a reference of the parent class type, although of course I won't be able to call any methods that are only declared in the derived class. But that's not a problem; I'm only calling a method that's declared in the parent class.

I have, however, added a `DoSomethingElse` method to the Parent class, and I've changed `ParentClass.DoSomething` to include a call to its own `DoSomethingElse` method.

When I create `myClass` as an instance of `ParentClass`, the output is what you'd expect:

```
ParentClass.DoSomething
ParentClass.DoSomethingElse
```

But watch what happens when I create `myClass` as an instance of `ChildClass`:

```
ParentClass.DoSomething
ChildClass.DoSomethingElse
```

Now the `ParentClass.DoSomething` method does something magical: it doesn't call its own `DoSomethingElse` method, it calls the *derived DoSomethingElse method in the child class*. The mechanism, if you're interested, is something called a Virtual Method Table, or VMT. The runtime uses the VMT to determine if there's a derived method that should be called in place of the parent method.

This has profound implications for programming. Virtual methods make it possible to plug new functionality into an existing class *without changing how that class's methods are called*. The Clarion ABC library exploits this feature to enable embed points.

Let's say you want to set a hot key for a window, for some nefarious purpose known only to yourself. The canonical way to do this is to use the `WindowManager.SetAlerts` embed point.

If you look in `ABWINDOW.CLW` for the source to this method you'll see the following:

```
WindowManager.SetAlerts PROCEDURE
I UNSIGNED,AUTO
CODE
  LOOP I = 1 TO RECORDS(SELF.CI)
    GET(SELF.CL,I)
    SELF.CL.WC.SetAlerts
  END
  IF ~SELF.History &= NULL
    LOOP I = 1 TO RECORDS(SELF.History)
      GET(SELF.History,I)
      SELF.History.Control{PROP:Alrt,255} = SELF.HistoryKey
    END
  EN
```

This method loops through a couple of internal queues and sets up any standard alert keys.

When you add code to an embed point, what you're really doing is creating an analog of the `ChildClass.DoSomethingElse` procedure declared above. Figure 1 shows the embed point in the embeditor.


```

ThisWindow.SetAlerts PROCEDURE

! Start of "WindowManager Method Data Section"
! [Priority 5000]

! End of "WindowManager Method Data Section"
CODE
! Start of "WindowManager Method Executable Code Section"
! [Priority 2500]

! Parent Call
PARENT.SetAlerts
! [Priority 5800]

! End of "WindowManager Method Executable Code Section"

```

Figure 1. SetAlerts in the embeditor

Add some code in either Executable Code Section embed point and you'll end up with a method that looks like this:

```

ThisWindow.SetAlerts PROCEDURE

CODE

! Some of my code here

PARENT.SetAlerts

! Some more of my code here

```

Whether a call to SetAlerts exists in a derived ThisWindow class or in the base WindowManager class, the derived SetAlerts will always be called. The derived SetAlerts has to explicitly call the parent version (PARENT.SetAlerts) to ensure the parent code is run (which is why you frequently see embed points as being either before or after the parent call).

Next up: interfaces

Although inheritance is often thought of primarily as a way to achieve code reuse, it's also a useful technique for designing flexible, adaptable applications. But it's not the only way. You can also design a "plug and play" architecture around a class-like structure called an *interface*. I'll have more on interfaces [next time](#).

[David Harms](#) is an independent software developer and the editor and publisher of Clarion Magazine. He is also co-author with Ross Santos of *Developing Clarion for Windows Applications*, published by SAMS (1995), and has written or co-written several Java books. David is a member of the American Society of Journalists and Authors ([ASJA](#)).

Reader Comments

[Add a comment](#)

Clarion Magazine

Writing To The Windows Registry

by Richard Rose

Published 2008-03-13

Okay how many times have you opened Clarion, clicked File|Open and then spent ages navigating to the folder you want because the shortcuts in the sidebar are just default ones that Microsoft feels you need?

Well, I don't like everything Microsoft does so I like to change things, and one of those is the sidebar, officially called the *places bar*, in the file dialog window. In this article I'll show you how to write code to change registry settings affecting this sidebar, and I'll also illustrate the use of arrays.

Custom place bar menus

If you're an MS Office user you may know you can modify sidebars already, but only for the MS Office file dialog windows; I want to make my custom sidebar menu appear when I use Clarion. Figure 1 shows the standard open file dialog as called from Clarion 6.

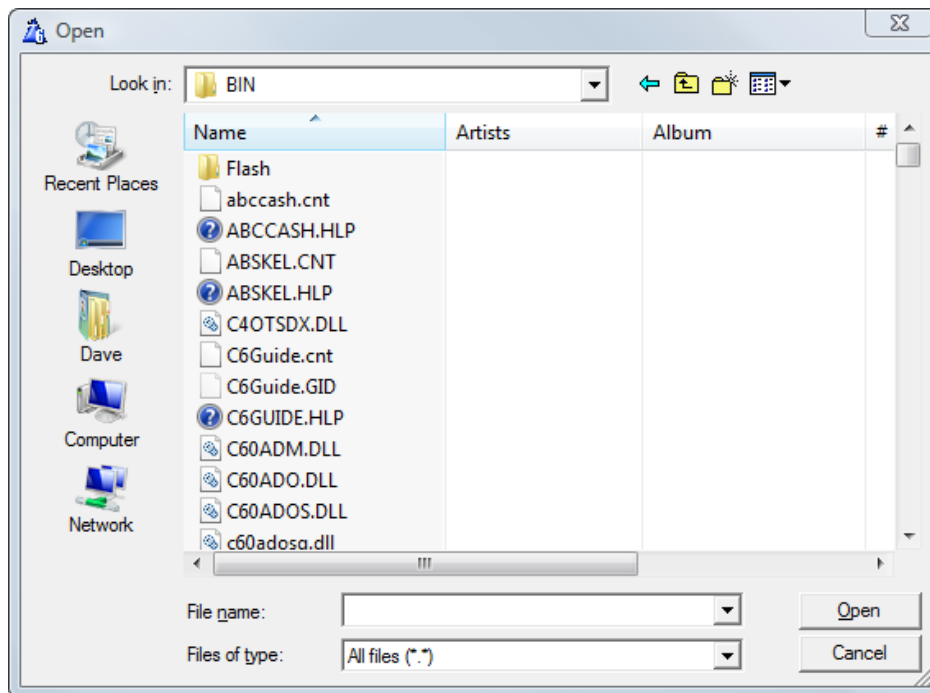


Figure 1. Opening a file from the Clarion IDE

NOTE: Opening a file from the Clarion IDE shows a similar dialog whether you're on XP or Vista. But applications designed for Vista (such as the C7/Clarion.NET IDE) will show a different dialog with a Favorite Links bar instead of a places bar. Favorite Links are easily modified by dragging and dropping, and are not the subject of this article.

Having located some [useful info](#) from the PuPpYpc.com website, I knocked together a quick and dirty app that lets me change the sidebar shortcuts.

What this app does is show you how to read and write registry settings. Although you can do this with a registry editor, it's often useful to make registry changes from your program. For instance, you might want to enforce Oplocks and cache settings to ensure TPS files run smoothly. If you do this every time your app runs (instead of relying on it being set once when you install using Friedrich Linder's excellent Setupbuilder or any other software installer) then you avoid the problem of some other software install undoing your settings!

The source code

The idea behind the Places application is simple: when the main (and only) window opens the code looks for the registry settings; if it finds the settings it populates the data into the corresponding entry field. When you save your changes, the app saves your settings back into the registry. Figure 2 shows the Places application as it will appear the first time you run it.

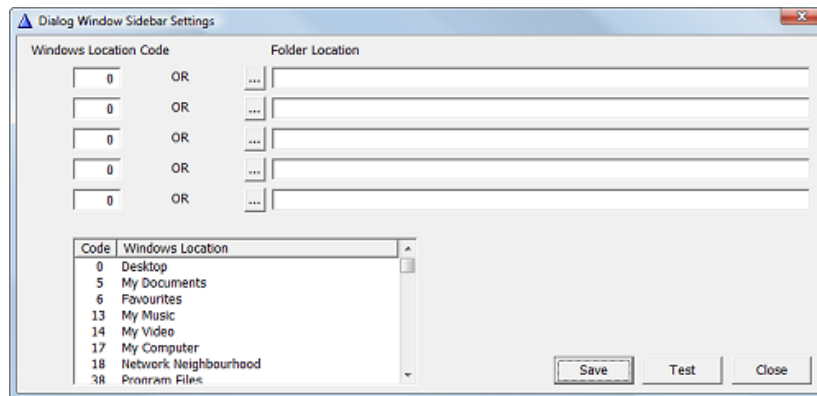


Figure 2. The Sidebar application ([view full size image](#))

Here's the code. First up are some local variables:

```
FolderLocation  STRING(256),DIM(5)
WindowsLocationCode  DECIMAL(2),DIM(5)
RegKey          STRING(256)
RegResult       STRING(256)
WindowsLocationCodeQ  QUEUE,PRE(WL)
Code            DECIMAL(2)
Description     STRING(30)
                END
Result         LONG
i              LONG
```

FolderLocation is an array of five strings to hold any specific directories for which you may want shortcuts. Similarly WindowsLocationCode contains up to five Windows codes indicating standard directories, such as Desktop = 0, My Documents = 5, My Computer = 17 and so on. You can see a list in the application, and in the source code as related to WindowsLocationCodeQ.

RegKey is the registry key, and RegResult is used to store existing values specified in the registry. Result is used to store return values from PutReg, and i is simply an index variable for looping through arrays.

Here's the code to retrieve any existing values:

```
i = 1
```

```

RegKey = |
'Software\Microsoft\Windows\CurrentVersion\Policies\comdlg32\PlacesBar'
Loop 5 times
RegResult = GETREG(REG_CURRENT_USER,RegKey,'Place' & i-1)
IF Len(Clip(RegResult)) > 2 !ie C:\ or longer
  WindowsLocationCode[i] = 0
  FolderLocation[i] = RegResult
Else
  WindowsLocationCode[i] = RegResult
  FolderLocation[i] = ""
END
i += 1
End
Display

```

This code uses GetReg with the specified key to retrieve each of the five key values; if the value is more than two characters long it is a directory (FolderLocation), otherwise it is a standard location code (WindowsLocationCode).

When you save your changes, the following code updates the registry.

```

i = 1
RegKey = |
'Software\Microsoft\Windows\CurrentVersion\Policies\comdlg32\PlacesBar'
Loop 5 times
DeleteReg(REG_CURRENT_USER,RegKey,'Place' & i-1)
If Len(Clip(FolderLocation[i])) > 0
  result = PUTREG(REG_CURRENT_USER,RegKey,|
  'Place' & i-1,Clip(FolderLocation[i]),REG_SZ)
Else
  result = PUTREG(REG_CURRENT_USER,RegKey,|
  'Place' & i-1,WindowsLocationCode[i],REG_DWORD)
End
If result <> 0
  Message('Error updating registry: ' & result)
End
i += 1
End

```

The code cycles through the array fields and then deletes the registry entry before putting a new one there. Although the delete may not be strictly necessary, I prefer to do it this way to ensure I'm cleaning up any old entries.

If FolderLocation has a value then that value is saved into the Placex registry key as a String (Reg_SZ); otherwise WindowsLocationCode is saved to the same Placex as a DWORD.

Now remember, when working with MS they often start array indexes at zero while Clarion array indexes start at 1 (the exception being Clarion#), so when the code creates the Placex values it uses the index value - 1. That is, the first entry on the window will be stored as Place0 in the registry.

You might argue arrays are not necessary here as you can achieve the same thing by storing the data in a record instead. You'd be right, but arrays are quick as they don't have the file access overhead; if you want the fifth record in an array you

just use MyArrayField[5] and that's it, you have the value.

While arrays are quick, don't expect to store thousands or millions of records in them. Large arrays use up memory very quickly; there's a reason why page loading browsers exist!

NOTE: If you're running Vista and you have UAC enabled, you'll need to run Sidebar.exe as an administrator. If you don't, the registry updates will fail.

Summary

Well that's a brief look at working with the Windows Registry and at the same time you get a nifty little utility which could help navigating to your favourite locations quicker in Clarion. And here's a tip: if you want more than five shortcuts, specify one of your five shortcuts as a folder containing additional shortcuts. That way you're still only two mouse clicks away from your favorite links.

[Download the source](#)

Reader Comments

Posted on Thursday, March 13, 2008 by Rick Martin

Richard,

This is totally brilliant!

I've setup a new folder with all of my common project and Clarion related folders and added that to the Places Bar.

Pretty much two clicks away (at most) from my most frequently accessed files.

Great!

Thanks,

Rick

[Add a comment](#)

Clarion Magazine

Clarion Magazine Readers In 121 Countries

by Dave Harms

Published 2008-03-12

In the wide world of programming languages, Clarion is clearly a niche product. And nobody (except maybe SoftVelocity) really seems to know just how many of us there are out there.

I can't, with certainty, answer the question of the size of the Clarion community, but after doing some digging in the Clarion Magazine database I am able to provide some information on the distribution of Clarion developers worldwide. In this article I'll present that information, and I'll briefly discuss the SQL I used to extract the data from Clarion Magazine's MySQL database.

To compile this information I looked at all registration data in Clarion Magazine where registrants had named their city. I used that particular criterion because it's an indicator that the country information was correctly set (it's possible to register with Clarion Magazine without giving detailed address information). Also note that these are registrants, who may or may not be subscribers. A free membership in Clarion Magazine gets you access to a variety of articles.

Figure 1 shows the distribution for the top 19 countries, with the remaining countries grouped under "Other" for a total of 20 wedges in the pie.

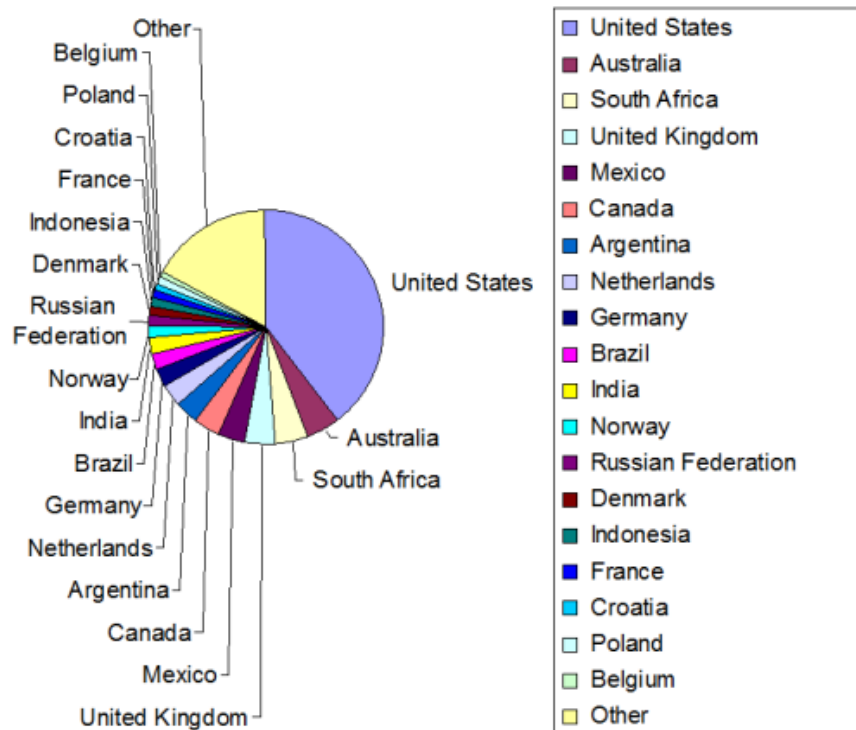


Figure 1. Top 19 countries

It will come as no surprise that the United States is home to the single largest contingent of Clarion developers.

Australia (at 4.62%) has long been in the number two spot at ClarionMag, which I find quite interesting. A few years back I heard a lot of grumbling from some Aussie developers about the dim future for Clarion down under. Despite that, Australia is the only place in the world right now with an [annual Clarion DevCon](#). Go figure.

South Africa (4.58%), long a hotbed of Clarion development, is always a lock for one of the top spots. And there's a decent population of Clarion developers in the UK as well. Mexico comes in at a surprising fifth place, ahead of Canada.

In terms of the number of Clarion developers, I suspect that Argentina should be much higher in the list than number seven. There is a very large population of Spanish-speaking Clarion developers in South America (and they've had a number of DevCons in recent years), but exchange rates and language barriers make it difficult for them to participate in Clarion Magazine. I've investigated various options for providing Spanish translations of the magazine, but unfortunately none of these have so far made economic sense.

Eastern and Western European countries make up the remainder of the pie chart, with the exception of Brazil, India, and Indonesia.

The following table shows all 121 countries represented in the Clarion Magazine database.

1	United States	39.50%
2	Australia	4.60%
3	South Africa	4.60%
4	United Kingdom	4.10%
5	Mexico	3.60%
6	Canada	3.60%
7	Argentina	3.40%
8	Netherlands	3.10%
9	Germany	2.30%
10	Brazil	2.20%
11	India	2.20%
12	Norway	1.70%
13	Russian Federation	1.40%
14	Denmark	1.30%
15	Indonesia	1.20%
16	France	1.10%
17	Croatia	1.00%
18	Poland	1.00%
19	Belgium	0.80%
20	New Zealand	0.80%

21	Spain	0.70%
22	Malaysia	0.70%
23	Yugoslavia	0.70%
24	Slovenia	0.60%
25	Colombia	0.60%
26	Switzerland	0.60%
27	Philippines	0.50%
28	Venezuela	0.50%
29	Singapore	0.50%
30	Sweden	0.40%
31	Finland	0.40%
32	Greece	0.40%
33	Pakistan	0.40%
34	Turkey	0.40%
35	Italy	0.40%
36	Puerto Rico	0.40%
37	Lithuania	0.30%
38	Hong Kong	0.30%
39	Ireland	0.30%
40	Korea	0.30%
41	Uruguay	0.30%
42	Algeria	0.30%
43	Dominican Republic	0.30%
44	Ukraine	0.30%
45	Austria	0.20%
46	Paraguay	0.20%
47	Tunisia	0.20%
48	Honduras	0.20%
49	Bosnia	0.20%
50	Chile	0.20%

51	Saudi Arabia	0.20%
52	Angola	0.20%
53	China	0.20%
54	Iceland	0.20%
55	Iran (Islamic Republic of)	0.20%
56	Morocco	0.20%
57	Peru	0.20%
58	Zimbabwe	0.20%
59	Ghana	0.10%
60	United Arab Emirates	0.10%
61	Costa Rica	0.10%
62	Czech Republic	0.10%
63	Egypt	0.10%
64	Hungary	0.10%
65	Kenya	0.10%
66	Nigeria	0.10%
67	Viet Nam	0.10%
68	Albania	<0.1%
69	American Samoa	< 0.1%
70	Andorra	< 0.1%
71	Bahamas	< 0.1%
72	Barbados	< 0.1%
73	Belize	< 0.1%
74	Benin	< 0.1%
75	Bermuda	< 0.1%
76	Bolivia	< 0.1%
77	Botswana	< 0.1%
78	Bulgaria	< 0.1%
79	Burkina Faso	< 0.1%
80	Cyprus	< 0.1%

81	Ecuador	< 0.1%
82	Faroe Islands	< 0.1%
83	French Polynesia	< 0.1%
84	Futuna Islands	< 0.1%
85	Georgia	< 0.1%
86	Greenland	< 0.1%
87	Guadeloupe	< 0.1%
88	Guinea	< 0.1%
89	Haiti	< 0.1%
90	Israel	< 0.1%
91	Jamaica	< 0.1%
92	Jan Mayen Islands	< 0.1%
93	Japan	< 0.1%
94	Kazakhstan	< 0.1%
95	Korea (Democratic)	< 0.1%
96	Latvia	< 0.1%
97	Lesotho	< 0.1%
98	Luxembourg	< 0.1%
99	Malta	< 0.1%
100	Mauritius	< 0.1%
101	Monaco	< 0.1%
102	Mongolia	< 0.1%
103	Myanmar	< 0.1%
104	Namibia	< 0.1%
105	Nepal	< 0.1%
106	Netherlands Antilles	< 0.1%
107	Nevis	< 0.1%
108	Nicaragua	< 0.1%
109	Northern Mariana Islands	< 0.1%
110	Panama	< 0.1%

111	Portugal	< 0.1%
112	Reunion	< 0.1%
113	Romania	< 0.1%
114	Senegal	< 0.1%
115	Sri Lanka	< 0.1%
116	Taiwan	< 0.1%
117	Thailand	< 0.1%
118	Trinidad	< 0.1%
119	Uzbekistan	< 0.1%
120	Vanuatu	< 0.1%
121	Zaire	< 0.1%

There's some interesting data there. In particular, I'm fascinated by the island nations, although for year round living I'd sooner set up shop in French Polynesia than in Norway's Jan Mayen Island in the Arctic Ocean. Or how about Reunion, in the Indian Ocean east of Madagascar? Looks like a nice place for a Euro DevCon; Reunion is an overseas region of France and therefore part of the EU. Or perhaps the mysterious island shown in Figure 2 would be more appropriate. What do you think?



Figure 2. Future Clarion DevCon location?

And a little SQL

For those of you interested in the SQL behind this analysis, here's the statement I used:

```
select count(*) * 100 / (select count(*)
```

```

from Names n where n.city <> ") as Count,
n.Country,c.Name from Names n left join ISOCountryCode c
on n.Country=c.Code2C
where n.City <> " group by n.Country
order by Count desc,c.Name;

```

The idea behind this statement is pretty simple: count the occurrences of each country and calculate that value as a percentage of the total number of records.

I'll break the SQL down into a series of tasks.

First, count the records. You don't need to know much about the Names table other than that it has Country and City fields:

```
select count(*),country from Names group by country;
```

If I want to order this list from highest to lowest, I write it this way:

```
select count(*) as count,country from Names
group by country order by count;
```

And if I want the list in descending order, with the highest number first:

```
select count(*) as count,country from Names
group by country order by count desc;
```

Whoops, one more thing. I said earlier that I only wanted to count records where there was a value for the City field:

```
select count(*) as count,country from Names
where city <> "
group by country order by count desc;
```

Now I'll add in the JOIN statement to retrieve the actual country names instead of just their two character codes (and I'll also subsort by the country name where the count is identical). ISOCountryCode is a list of, you guessed it, ISO country codes. I have both two character (Code2C) and three character (Code3C) values, and I need both for interfacing to different outside systems. Name is the name of the country (not to be confused with the Names table):

```
select count(*) as Count,Country,Name from Names
left join ISOCountryCode on Country=Code2C
where City <> " group by Country
order by Count desc,Country;
```

Now that I'm using two tables in my statement, there's some potential for confusing which fields (columns) go with which tables. For instance, I have a field called Name and a table called Names. Are they related? To which table does Country belong?

The solution is to use table prefixes. I change

```
Names
```

```
to
```

Names n

and

ISOCountryCode

to

ISOCountryCode c

Now my statement looks like this:

```
select count(*) as Count,n.Country,c.Name from Names n
left join ISOCountryCode c on n.Country=c.Code2C
where n.City <> " group by n.Country
order by Count desc,c.Name;
```

Okay, I'm almost home. The last thing I want to do is calculate a percentage value instead of a record count.

Fortunately, MySQL supports subselects, which are SELECT statements inside SELECT statements. I use the following to get the record total:

```
select count(*) from Names n where n.city <> ";
```

Now I want to change my record count to a percentage calculation. In pseudocode:

```
select count(*) * 100/ (get total number of records) as Count
```

To do this I enclose my subselect statement in parentheses and plug it into the SELECT statement. The result looks like this (with the subselect in italics):

```
select count(*) * 100/
(select count(*) from Names n where n.city <> ")
as Count,n.Country,c.Name from Names n
left join ISOCountryCode c on n.Country=c.Code2C
where n.City <> " group by n.Country order by Count desc,c.Name;
```

That statement gets me the results I'm looking for: a list of country names, in descending order, with percentage values to two decimal places. It's a little bit expensive in that the subselect has to run once for each city. Total execution time is 2.3 seconds. If I substitute a constant for the subselect execution time drops to 1.7 seconds, so clearly getting the total number of records isn't the biggest part of the job.

Countries I haven't heard from

So far I've been concerned with countries that are represented in the Clarion Magazine database. But what about the countries for which I don't have any valid Names records? This statement will find them by joining the Names table to the ISOCountryCode table and finding only those ISOCountryCode records which don't have a match:

```
select c.Name from ISOCountryCode c
left join Names n on c.Code2C=n.Country
where n.Country is null order by c.Name;
```

And here's the list:

- Burundi
- Cambodia
- Cape Verde
- Cayman Islands
- Central African Republic
- Chad
- Christmas Island
- Cocos (Keeling) Islands
- Comoros
- Congo
- Cuba
- Dominica
- East Timor
- Equatorial Guinea
- Ethiopia
- Falkland Islands (Malvinas)
- French Southern Territories
- Gabon
- Gambia
- Grenada
- Guam
- Guatemala
- Guinea-Bissau
- Guyana
- Heard & McDonald Islands
- Kiribati
- Kyrgystan
- Lao
- Lebanon
- Liberia
- Libyan Arab Jamahiriya
- Liechtenstein
- Macau
- Macedonia (former Yugoslavia)
- Madagascar
- Malawi
- Maldives
- Marshall Islands

- Martinique
- Mayotte
- Montserrat
- Mozambique
- Nauru
- Niger
- Niue
- Norfolk Island
- Palau
- Papua New Guinea
- Pitcairn
- Rwanda
- Saint Lucia
- Saint Vincent & Grenadines
- Samoa
- San Marino
- Sierra Leone
- Solomon Islands
- Sudan
- Suriname
- Tajikistan
- Tanzania
- The Grenadines
- Togo
- Tonga
- Turkmenistan
- Tuvalu
- US Minor Outlying Islands
- Virgin Islands (British);
- Western Sahara
- Yemen

It looks like there's still some room for Clarion to grow its worldwide market share. Anybody want to start up a Clarion user group on [Pitcairn Island](#)?

[David Harms](#) is an independent software developer and the editor and publisher of Clarion Magazine. He is also co-author with Ross Santos of *Developing Clarion for Windows Applications*, published by SAMS (1995), and has written or co-written several Java books. David is a member of the American Society of Journalists and Authors ([ASJA](#)).

Reader Comments

Posted on Friday, March 14, 2008 by Rex Kersley

The tropical island you picked looks as if it is Samoa. If it is, there may not be any Clarion programmers over there, but there are Clarion programs.

Cheers, Rex

Posted on Friday, March 14, 2008 by Dave Harms

Rex,

Ah, your programs, are they? Cool.

Nope, this one's quite a bit smaller than Samoa. If you'll think about it for a minute, you'll probably come up with the answer<g>. What's odd is I noticed it by complete coincidence the other night. We have one of those tabletop globes and I just happened to notice a teeny tiny speck and some teeny tiny print, somewhere off the west coast of one of the top five countries...

Dave

Posted on Friday, March 14, 2008 by douglas johnson

Similar to Reunion, but much closer travel for the editor. How about a DevCon to the east of ClarionMag?

Posted on Friday, March 14, 2008 by Dave Harms

Douglas,

Was that a suggestion (in which case I need another clue<g>) or a guess?

If a guess, here's a clue for you: this place is on approximately the same latitude as Mauna Kea.

Dave

Posted on Friday, March 14, 2008 by douglas johnson

> Was that a suggestion (in which case I need another clue<g>)
> or a guess?

One might consider it a suggestion via Juliet Binoche.

As for a guess, I think Ruy López de Villalobos MUST be added to everyone's list of favorite explorers.

Posted on Friday, March 14, 2008 by Dave Harms

> One might consider it a suggestion via Juliet Binoche.

Chocolate was a delight, but I'm no closer to the island. Guess I'm not too good at pop culture references<g>.

> As for a guess, I think Ruy López de Villalobos MUST
> be added to everyone's list of favorite explorers.

Ruy is da man!

Dave

Posted on Friday, March 14, 2008 by douglas johnson

>Chocolate was a delight, but I'm no closer to the island.

No, but St. John's is very close. A lasting gift from Jacques Cartier to the motherland.

Posted on Friday, March 14, 2008 by Dave Harms

I did wonder if you meant PEI. I've only been there once, before they built the bridge, and when the ferry came close and I saw those red clay cliffs I really wondered how it kept from just washing into the gulf.

It's a pretty place.

Dave

Posted on Friday, March 14, 2008 by Michael Brooks

What was your sample size Dave ?

Posted on Friday, March 14, 2008 by douglas johnson

> I did wonder if you meant PEI.

Not PEI. I was thinking Saint-Pierre (& Miquelon) - the last frontier of France in North America - where Juliet was "The Widow of Saint-Pierre"(Recommended) and the EU rules.

Posted on Friday, March 14, 2008 by Dave Harms

The last frontier of France, the next frontier of Canada!

Dave

Posted on Friday, March 14, 2008 by Dave Harms

Michael,

A closely guarded secret<g>

Dave

[Add a comment](#)

Clarion Magazine

The ClarionMag Blog

Get automatic notification of new items! [RSS feeds](#) are available for:

 [All blog entries](#)

 [All new items, including blogs](#)

Blog Categories

- [»All Blog Entries](#)
- [»Clarion 7 Clarion.NET](#)
- [»Future Articles](#)
- [»News flashes](#)
- [»Nifty Stuff](#)

I'm a hero!

Direct link

Posted Thursday, March 27, 2008 by Dave Harms

I'm just back from Microsoft Canada's [Heroes happen {here}](#) event in Winnipeg. I took in the developer session (presented by Microsoft's Jean-Luc David), which mainly dealt with Visual Studio 2008. Overall it was a good presentation, although as is the case with these kinds of events you seldom get to see code in much detail, and everything goes by in a blur. Rather than give you a blow by blow report, I'll focus on the things that stuck in my mind.

[LINQ](#) got a fair bit of coverage. LINQ stands for Language INtegrated Query, and is another fascinating product to come out of the C# team headed up by Anders Hejlsberg, the man largely responsible for Turbo Pascal and Delphi. (Some say there's a Clarion connection to Hejlsberg - certainly there's one via Niels Jensen, one of the founders of Borland who later left to form JPI, which merged with Clarion to become TopSpeed Corporation.)

LINQ is pretty cool stuff, and I expect it will become the standard .NET query language before too long. The big deal with LINQ is you can use it to query a lot of different kinds of data. You can query SQL, of course, but you can also use it with XML and all kinds of lists and collections. Just as you have ADO.NET providers (think drivers), so you have LINQ providers for databases such as Oracle, PostgreSQL, and MySQL. But don't get hung up on just databases. How about [querying Amazon](#), or [Flickr](#)?

I've been tracking LINQ for a while, but I'm less familiar with Windows Presentation Foundation (WPF). There are any number of ways to create user interface elements in .NET, and WPF (formerly Avalon) aims to be the "one ring to rule them all." WPF produces vector-based graphics, not raster graphics, so the idea is you can scale user interface

widgets under WPF. So how do you build WPF applications? You use a markup language called XAML (pronounced "zammel") to assemble the UI and wire up the business logic to that UI.

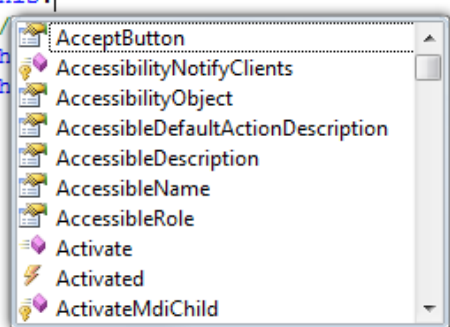
The visual scalability of XAML/WPF apps is evident in the Visual Studio XAML editor; there's a slider control with which you can resize the window and its controls. Depending on the number of controls, you might scale down all or part of a desktop window so it could fit on a mobile device.

How about WPF on the web? That's where Microsoft's Silverlight comes in - it's a way to present XAML apps in a web browser.

The final item before lunch was a reporting demonstration, but David had difficulties connecting to SQL Server. These connection problems were responsible for a number of glitches in the presentation; you gotta love demos. We did get to see a report wizard after the lunch break (using an Access database) and after many years of enjoying (if that's the right word) Clarion's extensive reporting capabilities I have to say I found the reporting demo singularly unimpressive. But at least you no longer need Crystal Reports in Visual Studio.

And here's a nicety in VS: when the Intellisense window is displayed, it may cover some of your code. Pressing Ctrl fades the window so you can see the code underneath. Here's the usual Intellisense window:

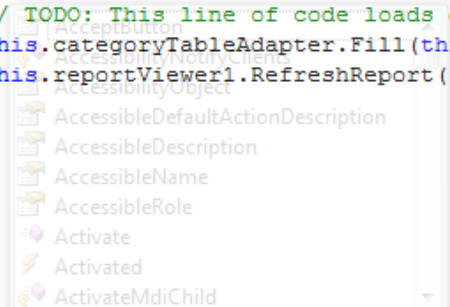
```
private void Form1_Load(object sender, EventArgs e)
{
    this.
//
th
th
}
// data into the
is.monorailDat
);
```



The screenshot shows the Intellisense window in Visual Studio. The window is a list box containing various accessibility-related items, including 'AcceptButton', 'AccessibilityNotifyClients', 'AccessibilityObject', 'AccessibleDefaultActionDescription', 'AccessibleDescription', 'AccessibleName', 'AccessibleRole', 'Activate', 'Activated', and 'ActivateMdiChild'. The window is partially overlapping the code in the background.

And here's what it looks like when I press Ctrl:

```
private void Form1_Load(object sender, EventArgs e)
{
    this.
// TODO: This line of code loads data into the
this.categoryTableAdapter.Fill(this.monorailDat.
this.reportViewer1.RefreshReport ();
}
// data into the
is.monorailDat
);
```



The screenshot shows the Intellisense window in Visual Studio, but it is faded. The same list of accessibility-related items is visible, but they are semi-transparent, allowing the code in the background to be seen more clearly.

You might have to squint a bit - the window is seriously faded. As soon as I release Ctrl the Intellisense window is back to its usual appearance.

Sometimes it's the little things that matter so much.

The lone afternoon session began with a lengthy and sometimes quite funny commercial/video, featuring a psychiatrist counseling a user who had lost his first love for his PC. He rediscovers his passion upon visiting a thinly-disguised Apple store, which he derides as a "candy coated wonderland of faux artsiness."

The rest of the session dealt with web development, including a described (but not demonstrated) web control which creates forms and views based on the structure of the database (rather than a data dictionary). There were some nifty demonstrations of web services courtesy of Windows Communication Foundation (WCF, formerly Indigo), which I didn't entirely follow. But it appears that among other things WCF makes it easy to create one service which is available in multiple dialects, including SOAP, REST, JSON, and POX (plain old XML). And if you're accustomed to FireFox's web development addons, take a look at [Web Development Helper](#) for IE.

Visual Studio's ability to debug Javascript looks pretty useful; if you've done any amount of web development, you're either already using Javascript or you will soon.

There are also a couple of nifty new web controls, including ListView, which allows you to customize the HTML used to display the data, and DataPager, which makes paginated browses easy to create. These are both new in .NET 3.5.

David answered the oft-asked question of which of Microsoft's many web-related tools developers should actually use. The short answer:

- Use [Expression Web](#) to for the visual aspect of regular web applications
- Use [Sharepoint Designer](#) for the visual aspect of XAML/WPF applications
- Use [Visual Studio](#) for all the back end code/business logic
- Forget about FrontPage - it's going away

On the subject of separating the user interface from the business logic, Microsoft's new [ASP.NET MVC framework](#) got a mention. MVC stands for [Model-View-Controller](#) and has been around for almost thirty years. In the last decade or so it's become more popular for web applications (ClarionMag is delivered to you by way of a custom MVC application) and with Microsoft finally getting on the bandwagon MVC appears to be going mainstream. A preview release of ASP.NET MVC is available, and requires .NET 3.5.

One final note on web development: if you're still using tables to format your web pages you need to get down and dirty with cascading style sheets, or CSS. For an extreme example of how you can radically change the appearance of a page with style sheets, and without altering the page itself in any way, check out the [CSS Zen Garden](#) and click on the alternate design links. Each link simply applies a different style sheet to the page you're viewing.

Of course, it wouldn't be a proper Microsoft seminar without some treats from Bill. The goodie bag included an NFR version of Visual Studio 2008 Standard Edition, the November 2007 CTP for SQL Server 2008 with a voucher for an eval version of Standard Edition when released, one year trial editions of Windows Server 2008 and Vista Ultimate SP1, plus a few other lesser items. If you can't get to a seminar, or you don't want to shell out for Visual Studio, you can still get the [individual express versions](#) for free.

I don't use Visual Studio that much, but I do find it handy to keep around as a learning tool and a reference point. I sometimes find myself compiling (and occasionally writing) C# code so I can be sure it works as expected; mostly I do this in Clarion.NET using the C# compiler, but certainly VS is the gold standard for hand coding, especially when it comes to the latest and greatest .NET libraries. For more on Clarion# as compared to Visual Studio check out the [Clarion.NET FAQ](#).

If you get a chance to take in a similar seminar, I think you'll probably find it worth your while. I certainly gained a helpful perspective on some of Microsoft's new technology.

Template Language Functions

[Direct link](#)

Posted Thursday, March 27, 2008 by Dave Harms

In his [articles this week](#) on EVALUATE, Paul Blais notes that not all Clarion language statements are available to evaluate. And that brings to mind another place where a subset of language functions is available, and that's within the template language.

Just as I was releasing Paul's article I came across a note Bob Foreman posted some time ago in the newsgroups (I don't recall just where or when), listing the Clarion language functions which can be used in conjunction with the template language. That is, you can use them in template expressions, making the template language an even more powerful code-generation tool. You can even use EVALUATE within template expressions, although BIND isn't available.

- . ABS
- . ACOS
- . ADDRESS
- . AGE
- . ALL
- . ASIN
- . ATAN
- . CENTER
- . CHOOSE
- . CHR
- . CLIP
- . CLOCK
- . COS
- . DATE
- . DAY
- . DEFORMAT
- . EVALUATE
- . FONTDIALOG
- . FORMAT
- . GETINI
- . INLIST
- . INRANGE
- . INSTRING
- . INT
- . LEFT
- . LEN
- . LOG10
- . LOGE
- . LONGPATH
- . LOWER
- . MATCH
- . MEMORY
- . MONTH

- NUMERIC
- PATH
- PUTINI
- RANDOM
- RIGHT
- ROUND
- SHORTPATH
- SIN
- SQRT
- STRPOS
- SUB
- TAN
- TODAY
- UPPER
- VAL
- YEAR
-

If you've never taken the time to look at the template language, you're missing out on one of the truly great things about Clarion programming. Check out the [template programming topic](#), in particular the Understanding Clarion Templates series.

Clarion Magazine

I'm a hero!

Published 2008-03-27

Blogs on this site

- »All Blog Entries
- »Clarion 7 Clarion.NET
- »Future Articles
- »News flashes
- »Nifty Stuff

I'm just back from Microsoft Canada's [Heroes happen {here}](#) event in Winnipeg. I took in the developer session (presented by Microsoft's Jean-Luc David), which mainly dealt with Visual Studio 2008. Overall it was a good presentation, although as is the case with these kinds of events you seldom get to see code in much detail, and everything goes by in a blur. Rather than give you a blow by blow report, I'll focus on the things that stuck in my mind.

[LINQ](#) got a fair bit of coverage. LINQ stands for Language INtegrated Query, and is another fascinating product to come out of the C# team headed up by Anders Hejlsberg, the man largely responsible for Turbo Pascal and Delphi. (Some say there's a Clarion connection to Hejlsberg - certainly there's one via Niels Jensen, one of the founders of Borland who later left to form JPI, which merged with Clarion to become TopSpeed Corporation.)

LINQ is pretty cool stuff, and I expect it will become the standard .NET query language before too long. The big deal with LINQ is you can use it to query a lot of different kinds of data. You can query SQL, of course, but you can also use it with XML and all kinds of lists and collections. Just as you have ADO.NET providers (think drivers), so you have LINQ providers for databases such as Oracle, PostgreSQL, and MySQL. But don't get hung up on just databases. How about [querying Amazon](#), or [Flickr](#)?

I've been tracking LINQ for a while, but I'm less familiar with Windows Presentation Foundation (WPF). There are any number of ways to create user interface elements in .NET, and WPF (formerly Avalon) aims to be the "one ring to rule them all." WPF produces vector-based graphics, not raster graphics, so the idea is you can scale user interface widgets under WPF. So how do you build WPF applications? You use a markup language called XAML (pronounced "zammel") to assemble the UI and wire up the business logic to that UI.

The visual scalability of XAML/WPF apps is evident in the Visual Studio XAML editor; there's a slider control with which you can resize the window and its controls. Depending on the number of controls, you might scale down all or part of a desktop window so it could fit on a mobile device.

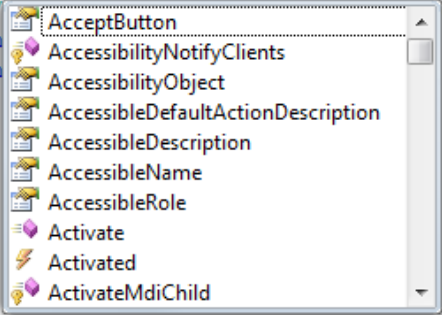
How about WPF on the web? That's where Microsoft's Silverlight comes in - it's a way to present XAML apps in a web browser.

The final item before lunch was a reporting demonstration, but David had difficulties connecting to SQL Server.

These connection problems were responsible for a number of glitches in the presentation; you gotta love demos. We did get to see a report wizard after the lunch break (using an Access database) and after many years of enjoying (if that's the right word) Clarion's extensive reporting capabilities I have to say I found the reporting demo singularly unimpressive. But at least you no longer need Crystal Reports in Visual Studio.

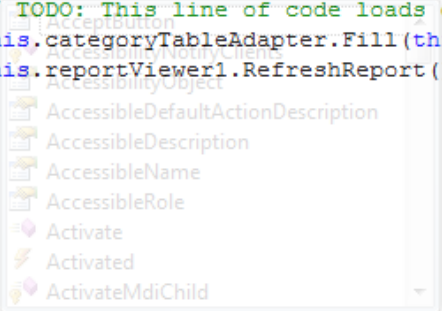
And here's a nicety in VS: when the Intellisense window is displayed, it may cover some of your code. Pressing Ctrl fades the window so you can see the code underneath. Here's the usual Intellisense window:

```
private void Form1_Load(object sender, EventArgs e)
{
    this.
    // data into the
    th is.monorailDat
    th );
}
```



And here's what it looks like when I press Ctrl:

```
private void Form1_Load(object sender, EventArgs e)
{
    this.
    // TODO: This line of code loads data into the
    this.categoryTableAdapter.Fill(this.monorailDat
    this.reportViewer1.RefreshReport ();
}
```



You might have to squint a bit - the window is seriously faded. As soon as I release Ctrl the Intellisense window is back to its usual appearance.

Sometimes it's the little things that matter so much.

The lone afternoon session began with a lengthy and sometimes quite funny commercial/video, featuring a psychiatrist counseling a user who had lost his first love for his PC. He rediscovers his passion upon visiting a thinly-disguised Apple store, which he derides as a "candy coated wonderland of faux artsiness."

The rest of the session dealt with web development, including a described (but not demonstrated) web control which creates forms and views based on the structure of the database (rather than a data dictionary). There were some nifty demonstrations of web services courtesy of Windows Communication Foundation (WCF, formerly Indigo), which I didn't entirely follow. But it appears that among other things WCF makes it easy to create one service which is available in multiple dialects, including SOAP, REST, JSON, and POX (plain old XML). And if you're accustomed to FireFox's web development addons, take a look at [Web Development Helper](#) for IE.

Visual Studio's ability to debug Javascript looks pretty useful; if you've done any amount of web development, you're either already using Javascript or you will soon.

There are also a couple of nifty new web controls, including [ListView](#), which allows you to customize the HTML used to display the data, and [DataPager](#), which makes paged browses easy to create. These are both new in .NET 3.5.

David answered the oft-asked question of which of Microsoft's many web-related tools developers should actually use. The short answer:

- Use [Expression Web](#) to for the visual aspect of regular web applications
- Use [Sharepoint Designer](#) for the visual aspect of XAML/WPF applications
- Use [Visual Studio](#) for all the back end code/business logic
- Forget about FrontPage - it's going away

On the subject of separating the user interface from the business logic, Microsoft's new [ASP.NET MVC framework](#) got a mention. MVC stands for [Model-View-Controller](#) and has been around for almost thirty years. In the last decade or so it's become more popular for web applications ([ClarionMag](#) is delivered to you by way of a custom MVC application) and with Microsoft finally getting on the bandwagon MVC appears to be going mainstream. A preview release of ASP.NET MVC is available, and requires .NET 3.5.

One final note on web development: if you're still using tables to format your web pages you need to get down and dirty with cascading style sheets, or CSS. For an extreme example of how you can radically change the appearance of a page with style sheets, and without altering the page itself in any way, check out the [CSS Zen Garden](#) and click on the alternate design links. Each link simply applies a different style sheet to the page you're viewing.

Of course, it wouldn't be a proper Microsoft seminar without some treats from Bill. The goodie bag included an NFR version of Visual Studio 2008 Standard Edition, the November 2007 CTP for SQL Server 2008 with a voucher for an eval version of Standard Edition when released, one year trial editions of Windows Server 2008 and Vista Ultimate SP1, plus a few other lesser items. If you can't get to a seminar, or you don't want to shell out for Visual Studio, you can still get the [individual express versions](#) for free.

I don't use Visual Studio that much, but I do find it handy to keep around as a learning tool and a reference point. I sometimes find myself compiling (and occasionally writing) C# code so I can be sure it works as expected; mostly I do this in [Clarion.NET](#) using the C# compiler, but certainly VS is the gold standard for hand coding, especially when it comes to the latest and greatest .NET libraries. For more on [Clarion#](#) as compared to Visual Studio check out the [Clarion.NET FAQ](#).

If you get a chance to take in a similar seminar, I think you'll probably find it worth your while. I certainly gained a helpful perspective on some of Microsoft's new technology.

Clarion Magazine

Template Language Functions

Published 2008-03-27

Blogs on this site

- [»All Blog Entries](#)
- [»Clarion 7 Clarion.NET](#)
- [»Future Articles](#)
- [»News flashes](#)
- [»Nifty Stuff](#)

In his [articles this week](#) on EVALUATE, Paul Blais notes that not all Clarion language statements are available to evaluate. And that brings to mind another place where a subset of language functions is available, and that's within the template language.

Just as I was releasing Paul's article I came across a note Bob Foreman posted some time ago in the newsgroups (I don't recall just where or when), listing the Clarion language functions which can be used in conjunction with the template language. That is, you can use them in template expressions, making the template language an even more powerful code-generation tool. You can even use EVALUATE within template expressions, although BIND isn't available.

- . ABS
- . ACOS
- . ADDRESS
- . AGE
- . ALL
- . ASIN
- . ATAN
- . CENTER
- . CHOOSE
- . CHR
- . CLIP
- . CLOCK
- . COS
- . DATE
- . DAY
- . DEFORMAT

- . EVALUATE
- . FONTDIALOG
- . FORMAT
- . GETINI
- . INLIST
- . INRANGE
- . INSTRING
- . INT
- . LEFT
- . LEN
- . LOG10
- . LOGE
- . LONGPATH
- . LOWER
- . MATCH
- . MEMORY
- . MONTH
- . NUMERIC
- . PATH
- . PUTINI
- . RANDOM
- . RIGHT
- . ROUND
- . SHORTPATH
- . SIN
- . SQRT
- . STRPOS
- . SUB
- . TAN
- . TODAY
- . UPPER
- . VAL
- . YEAR
- .

If you've never taken the time to look at the template language, you're missing out on one of the truly great things about Clarion programming. Check out the [template programming topic](#), in particular the Understanding Clarion Templates series.