

Clarion Magazine

Clarion News

- » [ClarioNET Demos Showcase Third Party Support](#)
- » [Data Manager 1.0.0.4](#)
- » [CoolFrames 1.14 Beta](#)
- » [iQ-Notes 3.60](#)
- » [Color Cop Adds Clarion Hex Output](#)
- » [Mario Wojcik Joins Huenuleufu](#)
- » [Clarion Developers Web Hosting Sale](#)
- » [FullRecord 2.00](#)
- » [FinalStep 2.16](#)
- » [New Data Management Utility \(Beta\)](#)
- » [Clarion Third Party Profile Exchange Update](#)
- » [Solace Software Templates/Products Update](#)
- » [RADFusion Acquires List and Label Templates](#)
- » [RADFusion Reforms As RADFusion LLC](#)
- » [vuFileTools Beta 3.4](#)
- » [Super QBE 6.62](#)
- » [RPMxt for C7.0 2509](#)
- » [SetupBuilder 6.6 Build 2016](#)
- » [Clarion Handy Tools Build 11D1.00](#)
- » [SV Blog: Clarion.NET Controls](#)
- » [New J-Media Demo](#)
- » [DCT2SQL Templates Updated](#)
- » [IceTips Magic Locks 1.1](#)
- » [Colin Wynn Heads UK User Group](#)
- » [Chicago CUG Oct Meeting](#)
- » [Tagkeys Uses Cool Frames](#)
- » [StrategyOnline Volume Discounts](#)
- » [J-Media 1.02](#)
- » [awDebugger 1.1](#)
- » [1stLogoDesign Fall Sale](#)
- » [SetupBuilder 6.6 Build 2000](#)
- » [Search Engine Profile Exchange Updated](#)
- » [Evolution Browse Export Professional 1.12](#)
- » [Clarion.NET Payment Gateway](#)
- » [Google Maps Example](#)

[\[More news\]](#)

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

Latest Free Content

- » [Source Code Library 2007.09.30 Available](#)

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

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



Latest Subscriber Content

Choosing Colors

Clarion applications are, by default, somewhat bland. And for most business applications that isn't necessarily a bad thing. Your clients probably buy your software because it does the job, not because it blasts them with vibrant color. But you may want or need a little color in your apps, and if not in your apps, then almost certainly on your web site. So how do you choose colors that work?

Posted Wednesday, October 31, 2007

A Simple Solution for Accordion Menus

David Podger and Deon Canyon are fans of third party Clarion products, and they could have chosen a commercial sidebar menu when the need arose. But the idea of writing a basic, button-oriented two-level menu had its appeal.

Posted Wednesday, October 31, 2007

Passing Variables To A Thread

What do you do when you have a process that creates a Word document, but you run into problems when using it in batch mode? You set up a background thread and pass it job parameters. Maarten Veenstra explains.

Posted Tuesday, October 30, 2007

Developer Bio: Ivan Grech Mintoff

Meet the new owner of Michael Brooks' ClarioNET. This long time Clarion user is a member of a politically prominent family on a famous Mediterranean island...

Posted Friday, October 19, 2007

Exporting Sub-App Prototypes With A Utility Template

Clarion developers often split applications up into main and sub-applications, benefiting team development and code reuse. But how can you easily ensure that prototypes remain synchronized? Dragan Duric presents a prototype exporting utility template.

Posted Thursday, October 18, 2007

Tiers and Objects: The Future of Clarion.NET Development

Is the move to the .NET platform a good thing or a needless complication? How will Clarion programming change? Will those changes benefit your customers?

Posted Thursday, October 11, 2007

PostgreSQL Revisited: Essential Tools

Dave Harms continues his look at PostgreSQL for Windows with a discussion of databases, tablespaces, and administrative tools.

Posted Thursday, October 11, 2007

Source Code Library 2007.09.30 Available

The Clarion Magazine Source Code Library has been updated to include the September source. Source code subscribers can download the Jan-September 2007 update from the [My ClarionMag](#) page. If you're on Vista please run Lindersoft's Clarion detection patch first.

Posted Monday, October 08, 2007

[\[Last 10 articles\]](#) [\[Last 25 articles\]](#) [\[All content\]](#)

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

Clarion Sites

» » [Atlantic Clarion User Group](#)

Clarion Blogs

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 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:

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

Clarion Magazine

Clarion News

[Search the news archive](#)

vuFileTools Newest Function

vuFileTools adds vuServiceStatus(ServerName,ServiceName) which returns the status of the service on (which by the way can be a client also). If you want to check a service on the PC that your app is running on, just leave blank. This is considered beta code.

Posted Tuesday, November 06, 2007

Data Manager Center 1.0.0.7

Data Manager Center 1.0.0.7 is now available. For those with or without a valid licence and with an older version than 1.0.0.3, it is recommended to download the full installer and install after removing the old version. If you have version 1.0.0.4 or above installed then please just use the updater from the application or the Program Group. As of 1.0.0.6 you can access any ORACLE ODBC table and transfer data from or to those tables. IDENTITY fields are not detected in Oracle - it will be up to you to be carefull about those. For this new Driver you will have a new field to enter the name of "your" local driver installed by Oracle - do not use the Microsoft one as it will not work and is slow in comparisson to the Oracle one. 1.0.0.7 lets you access any PostGreSQL ODBC Table and transfer data from or to those tables.

Posted Tuesday, November 06, 2007

Smart-type Technical Support Unavailable Nov 6-13

Smart-type technical support will be unavailable from Nov 6-13 as Dave Beggs is taking the family on a holiday to the beach.

Posted Tuesday, November 06, 2007

SetupBuilder 6.6 Build 2040

Lindersoft has released SetupBuilder 6.6 Build 2040. This release contains fixes that are designed to correct known issues and to improve the overall functionality of SetupBuilder. You can get the latest version by selecting "Check for Updates" from within the SetupBuilder 6 IDE. You can get the latest documentation by selecting "Check for Documentation Updates" from within the SetupBuilder 6 IDE.

Posted Tuesday, November 06, 2007

SV Blog Post On Moving To .NET

Bob Z blogs on the reasons for moving to .NET, including the familiarity of Clarion# code, language compatibility, data compatibility, multiple platform support, 64 bit apps, eye candy, the framework library, and more.

Posted Tuesday, November 06, 2007

Lodestar Products And C6.3 9057

All current C6 releases of Lodestar Products are compatible with C6.3 build 9057 and do not require any updates.

Posted Tuesday, November 06, 2007

Clarion Daily News Hits 10K

Clarion Daily News has hit 10000 visitors after 10 months online.

Posted Tuesday, November 06, 2007

ClarionFAQ Temporarily Down

Bad people have hacked the ClarionFAQ site. It's been taken down due to suspicious activity.

Posted Tuesday, November 06, 2007

Super Security 6.62

Super Security 6.62 is available for download from our site. The current list of C7-compatible installers is: Super QBE; Super Security; Super Stuff (MH); Super Tagging. Next up are Super Passcode and Super Limiter. After releasing C7-compatible updates for the rest of its products, BoxSoft will finalize the major changes to Super Invoice, and get it and the new Super Browse out the door. Then comes the major upgrade to Super Import-Export. For upgrades and the new passwords, please contact Mitten Software (www.mittensoftware.com)

Posted Tuesday, November 06, 2007

PostgreSQL Multi-Master Replication

Cybertec has announced the availability of an open source synchronous replication solution for PostgreSQL called Cybercluster. Cybercluster is able to keep an arbitrary number of database nodes in sync at any time and provides high-end replication technology for high-end and fault tolerant applications.

Posted Tuesday, November 06, 2007

ClarioNET Demos Showcase Third Party Support

Live demonstrations of ClarioNET are now available from the ClarioNET web server. Download the thin client to get access to several live ClarioNET server applications. The intention with the live demonstrations is to take the exact demo application that other third parties use to demonstrate their functionality and to make this application ClarioNET enabled. This of course is done with the permission of the vendor. If you're a third party vendor wishing to have a demo showcased please contact Ivan. There are also two short videos on how to implement ClarioNET in your pre-existing app or project, as well as a PowerPoint presentation on the benefits of providing ClarioNET solutions to your clients. Also note that the special price on ClarioNET ends Wed, October 31.

Posted Monday, October 29, 2007

Data Manager 1.0.0.4

Data Manager 1.0.0.4 is now available. Data Manager now has support for Clarion DAT files, as well as a Portuguese translation of the IDE. The web site is now fully multi-lingual; English and French are supported now with Spanish on the way. Release 1.0.0.4 fixes a bug found in the transfer of data when ODBC SQL was selected.

Posted Monday, October 29, 2007

CoolFrames 1.14 Beta

The CoolFrames 1.14 beta is now available, with new textured and gradient Vista looks and various bug fixes and improvements.

Posted Monday, October 29, 2007

iQ-Notes 3.60

iQ-Notes 3.60 is now available. This release adds a new Attach Note option. If you are an existing iQ-Notes user, just install it when iQ-Notes is closed.

Posted Monday, October 29, 2007

Color Cop Adds Clarion Hex Output

As of version 5.4.5, Color Cop has the option to display selected colors in Clarion's hex notation. Color Cop is a multi-purpose color picker for web designers and programmers. It features an eyedropper, magnifier, variable magnification levels, 3 by 3 and 5 by 5 average sampling, snap to websafe, color history, and a complementary color palette.

Posted Monday, October 29, 2007

Mario Wojcik Joins Huenuleufu

Mario Wojcik, an experienced Clarion programmer, has joined the Huenuleufu Development task force recently. You can see his photo in any of the About pages. Huenuleufu Development offers low cost development thanks to currency exchange rates.

Posted Monday, October 29, 2007

Clarion Developers Web Hosting Sale

From October 22 until October 31st Oak Park is running a sale on all of its shared hosting line of products. Products include Windows and Linux Hosting starting at \$3.99 per month, as well as storage and bandwith add-ons, dedicated IP addresses, Cold Fusion options, Online File Folders, Traffic Facts, and the Email line of products. Free private registration with five or more domain registrations or transfers. It was longer than usual since our last newsletter.

Posted Monday, October 29, 2007

FullRecord 2.00

FullRecord 2.00 is now available. Now every insert, update or delete operation is automatically audited, even if it is in source code. From this version on, FullRecord is compatible with Clarion 6 and later (Clarion 5.x is no longer supported). If you need to use FullRecord with a Clarion 5.5 program, you need to use the latest 1.x version for it. You may register both templates at the same time (of course, you can use only one at a time in a given app). If you don't own FullRecord, a new copy will cost US\$149; if you already own FullRecord 1.x and you do have your maintenance plan up to date, this upgrade will cost US\$49. If you already own FullRecord 1.x but you don't have a valid and current maintenance plan, this upgrade will cost you US\$99 for a limited time.

Posted Monday, October 29, 2007

FinalStep 2.16

FinalStep 2.16 is now available. Changes include: Fixed problem with run-time random wallpaper for frame; Fixed ABC support for C7 menus.

Posted Monday, October 29, 2007

New Data Management Utility (Beta)

The Data Management Utility, built with Clarion, lets you transfer data to and from a variety of databases without knowing anything of the structures. At present supported database drivers include BASIC (CSV), ODBC, and TPS, with more to come. You can use this tool in any direction (e.g. SQL to TPS or TPS to SQL). The utility will read all source and destination tables and offer you all the information you need. You can view the table structures and see all the data in

the source and destination tables. Mappings can be done automatically or manually, and you can define record and column level conditions. Profiles can be saved for later use. Free trial available. Price is €140, or approximately US\$200.

Posted Friday, October 19, 2007

Clarion Third Party Profile Exchange Update

To use this new Web Update method, download and install Product Scope 7. Then download and install a Web Updated Profile Exchange. You can check for updates via the Start Menu or the desktop icon.

Posted Friday, October 19, 2007

Solace Software Templates/Products Update

All Solace Software's templates have now gone to Russell Eggen at RADFusion. It's not clear yet what will happen to Data Ferret or Prospector but keep an eye on the DataFerret website.

Posted Friday, October 19, 2007

RADFusion Acquires List and Label Templates

RADFusion International is the new vendor for List and Label templates. These templates are built as wrappers around Combit's List and Label developer report tool. There is some work to be done to get these templates ready for the new version of List & Label. There will be more news about this soon. Expect upgrade prices to be on a par with Combit's history. If anyone has this product and has a long standing unresolved issue, please be sure to contact RADFusion (use the contact page if you don't have Russ's email address). Please list what version you have now (with serial number if you have it), what you'd like to see in a new release, any outstanding bugs, etc.

Posted Friday, October 19, 2007

RADFusion Reforms As RADFusion LLC

RADFusion Inc. has been reformed as RADFusion International, LLC, in keeping with its global expansion plans. The RADFusion domain will remain the same, no need to change any bookmarks. Same for all email contacts.

Posted Friday, October 19, 2007

vuFileTools Beta 3.4

vuFileTools Beta 3.4 is now available. There will be no charge for this update, and no increase in price (still \$49.00). There are a couple of changes that you need to be aware of before you download vuFileTools 3.4. A new .chm help file is available for download that hopefully helps with function grouping, etc., and is available for download; New passwords required - if you are a registered user of vuFileTools and want to download this beta release email Bill and he'll send you the new password as soon as he can (usually within 48 hrs). You must be a valid user to receive a new password. Some functions have changed! Be sure and read the help file. Some older versions of vuFileTools used to be installed directly under the root of Clarion. This install places everything the 3rdParty subdirectory. You will need to Unregister your current vuFileTools and re-register this one (don't erase your current version...this release is beta).

Posted Friday, October 19, 2007

Super QBE 6.62

Super QBE 6.62 is available for download. The current list of C7-compatible installers includes: Super QBE; Super Stuff (MH); Super Tagging. Next up is Super Security. After releasing C7-compatible updates for all of our existing products, Boxsoft will finalize the major changes to Super Invoice, and get it and the new Super Browse out the door. Then comes the upgrade to Super Import-Export. For upgrades and the new passwords, please contact Mitten Software.

Posted Friday, October 19, 2007

RPMxt for C7.0 2509

RPMxt for C7.0 2509 requires your C6 RPMxt upgrade unlock codes. Although legacy files are included, please note that extended target support is currently only for ABC applications. Legacy support will follow over the next few weeks. Updated installs for other Clarion 6 builds will be released soon. This release provides support for current SoftVelocity report targets and most, if not all, third party extended report targets. There's enough flexibility built into this release that you can even create your own targets, if you have the need, with ease. In addition this release will allow you to intercept the action before and after target generation which opens up all kinds of possibilities. A developer callback method is included so you can run any routine that's local to the report or any procedure that is available. This is accessible from the destination dialogs as well as the developer procedure. Page ranges, as has always been included in RPM, are supported for the extended targets. Also included is the option to restrict range support for individual targets. Programmatic forced target generation support is provided so that targets are processed immediately after the report has completed and before the user has access. Options to set target button text and icon as well as ability to add tool tips. Also included with this release is support for LOCAL compiles and CHM (compiled HTML help).

Posted Friday, October 19, 2007

SetupBuilder 6.6 Build 2016

Lindersoft has released SetupBuilder 6.6 Build 2016. Lindersoft strongly recommends customers upgrade to the latest version of SetupBuilder 6.6 as soon as possible to maintain the highest level of support, performance and reliability. If you have a current SetupBuilder Maintenance and Support Subscription Plan, the update is free of charge. You can get the latest version by selecting "Check for Updates" from within the SetupBuilder 6 IDE. Build 2016 is primarily a maintenance release that fixes reported issues and provides small feature enhancements.

Posted Friday, October 12, 2007

Clarion Handy Tools Build 11D1.00

Clarion Handy Tools has release build 11D1.00. It has been download-tested and compile-tested in both Windows XP and VISTA. Subscribers with current subscriptions please use your WEBUPDATER installer to download and install all new and revised classes, templates, demo applications and utilities.

Posted Friday, October 12, 2007

SV Blog: Clarion.NET Controls

Bob Foreman has been documenting the Clarion.NET WinForms controls; at present there are 48 listed in the (as yet unreleased) Clarion.NET IDE. Bob notes that there are over 800 WinForms controls available from the official MS WinForms web site.

Posted Friday, October 12, 2007

Clarion Magazine

Choosing Colors

by Dave Harms

Published 2007-10-31

Clarion applications are, by default, somewhat bland. And for most business applications that isn't necessarily a bad thing. Your clients probably buy your software because it does the job, not because it blasts them with vibrant color. But you may want or need a little color in your apps, and if not in your apps, then almost certainly on your web site.

So how do you go about choosing colors that are pleasing to the eye? If you don't have any interest or expertise in graphic design, your best option may be to hire a designer. But whether you decide to go it alone or hire an artist, you'll benefit from knowing a little bit about how computer monitors generate color, and how those colors are modeled. I'll cover these topics in this installment; in Part 2 I'll go on to principles of color selection and look at a few color picking tools.

Disclaimer

Although I have some interest in art and perhaps a modest level of ability, I'm no expert on colors. Just ask my wife. But over a decade of so of web development I've been forced to learn a few things about the nature of color, and it's that knowledge I'm passing along. I hope you will find it as helpful as I have. Just remember: if you want a really terrific user interface, whether for a desktop or a web site, the right graphic designer can make all the difference.

Color theory

Color theory is a vast subject with many competing and often conflicting theories. But the core of color theory, from an application/web site developer's standpoint, boils down to a few important points:

- Colors come in varying brightness
- Colors come in varying intensity (and yes, that is different from brightness)
- Colors have varying compatibility with each other; some combinations are pleasing and harmonious, some are conflicting and disturbing.

Okay, that's pretty much a minefield for us non-graphic designers, right? And I haven't even touched on the [emotional impact of color](#).

It really is extraordinarily easy to create unpleasant color schemes, just as it's easy to sit down at a piano and bash out an awful combination of notes. So here are a few concepts that are the color equivalent of learning to play Chopsticks.

Color models

The first thing to keep in mind about color is that producing color for printed material is quite different from producing color onscreen. In printed material you use various paints and inks containing pigments. These pigments absorb certain wavelengths of light, and the wavelengths that aren't absorbed are reflected, and that's the color you see. Pigment colors are subtractive - each pigment you add absorbs more color. That's why if you mix a random bunch of paints together you'll likely end up with a dark, muddy mess. On screen, however, colors are additive, not subtractive. They are typically made up of various combinations of red, green, and blue light; add these three colors together in equal amounts and you get white light.

Often you'll see the spectrum of colors displayed in a color wheel, as shown in Figure 1. Again, there are differences in color wheels depending on whether you're talking about print or online media. In both cases you have three primary colors, equally spaced around the color wheel. In painting these are blue, red, and yellow. The printing industry uses a similar color set: cyan, magenta, and yellow (CMYK - the K stands for black, the fourth "color"). Why cyan and magenta? Steve Parker informs me that the CMYK has its origins in color photography; cyan, not blue, is the color of the sky. On a computer monitor, as you've probably already guessed, the three primary colors are red, green, and blue.

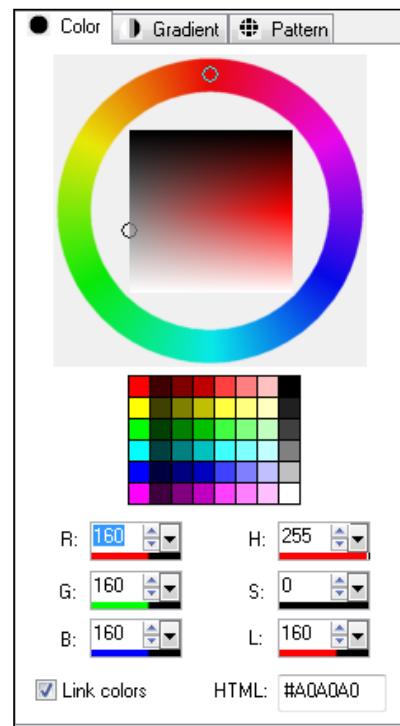


Figure 1. A color wheel and color picker (Paint Shop Pro)

Take a close look at the color wheel in Figure 1. The top of the wheel is red, a third of the way around clockwise you see blue, and another third of the way around is green. In most color pickers that use wheels you select a color at some point on the wheel, but since these are just the "pure" colors you need some other way of adjusting the color's brightness and intensity. I'll come back to that in a moment.

In Figure 1 you'll also see two sets of spin controls, one set labelled R, G, and B and the other labelled H, S, and L.

RGB you know - these allow you to set brightness levels for red, green and blue from 0 to 255.

HSL may, at first, seem a bit more confusing. H stands for hue, and again is a value from 0 to 255, representing 256 positions around the color wheel, 255 being red. S stands for saturation, which is the degree to which the specified color is diluted with gray; 0 is full color, 255 is completely gray. L stands for lightness (or luminosity). A lightness of 0 is full black, and a lightness of 255 is full white.

In the HSL model, a pure red, for instance, will have a hue of 255, a saturation of 255 (full color, no gray), and a lightness of 128 (halfway between white and black).

There is another model similar to HSL called HSV, for hue, saturation and value, which is more common on the Mac platform (and in Adobe tools).

Whichever model you use, you'll be dealing with some variation on the ideas of color, brightness, and degree of gray content.

Balanced colors

Colors are like fonts - in general, the fewer the better. The more colors you have in your application or on your page, the more likely your users are to be confused by the colors, and annoyed by disharmony among colors.

So how do you go about choosing harmonious colors? As with music, there are some basic rules which all beginners should follow (if you get good enough, with colors as with music, you can go ahead and break the rules).

Color wheels are often shown divided into sections that are some multiple of three (after the three primary colors). Twelve is a common number. Although twelve doesn't evenly divide into 256, you can get an approximation of 12 colors by incrementing (or decrementing) the hue in increments of 21 (21.3333 to be exact).



Figure 2. A twelve part color wheel

You can use a simple twelve part wheel (starting at any hue you like) to achieve some basic compatible color schemes. In each of these schemes, however, some combinations will work better than others. Note also that some combinations introduce a lot of contrast between colors. High contrast color schemes can be very exciting and dynamic, but as the rewards are higher so are the risks that the combination won't work. When dealing with highly contrasting colors it's often best to have one color clearly dominant.

Here are a few ways you can use a twelve part color wheel:

- Choose any three adjacent colors out of the twelve. These colors will be compatible because of their similarity, but you won't see a lot of contrast.
- Choose two opposite (complementary) colors. This is a high-contrast color scheme.
- Similar to the complementary scheme is one sometimes called split complementary. Instead of (or in addition to) one of the complementary colors, use the two colors adjacent to that color.
- Choose any three colors, evenly spaced around the color wheel. This is called a triadic color scheme. Triadic colors are less contrasty than complementary colors.
- Choose two pairs of complementary colors. While each pair will be in balance, it can be hard to find two pairs where all four colors are in balance.
- Use varying shades of a single color (hue, that is). You can achieve some interesting effects, although you won't get as vibrant a result as you will with multiple colors.
- Keep in mind that white and black can be just as important as colors - black, white, and a single color can make for a dramatic and harmonious look.

There are many other ways to choose compatible colors; these are just a few common techniques.

Once you've settled on a color scheme, you can start experimenting with saturation and lightness. Add in some gray (reduce saturation) to tone down the color(s) and reduce contrast; play with lightness to get the right light/dark balance. Just as you used a twelve part color wheel (in steps of approximately 21) you may want to take a similar, regular interval approach to saturation and lightness.

Sampling

Find an application (or a web site) with a look and feel you like and take a screen shot. Open it in your graphics editing software and use the eyedropper tool (or equivalent) to examine the hue and saturation levels of various design elements. You'll probably be surprised at just how much gray is in some colors, particularly those pastel shades. Sampling colors you like can be a great help in choosing your own color scheme.

Next time

Choosing colors can be hard work. You may find it difficult to come up with a color scheme you, and more importantly your customers, like (keeping in mind that you're never going to make everyone happy). Fortunately there are a number of color picking tools around that make this job a little bit (if not a lot) easier, and I'll cover some of those 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 Monday, November 05, 2007 by David Podger

Glad to see an article on this subject. I hope you may show, in your sequel, how to generate a small set of suitable colours from a given, starter colour using the different principles involved. That is, how to do this in Clarion given just one Hex starter colour. Thanks again,

.....
Posted on Monday, November 05, 2007 by Dave Harms

David,

There are definitely ways to do that, and I'll cover a bunch of tools in Part 2.

Dave

[Add a comment](#)

Clarion Magazine

A Simple Solution for Accordion Menus

by David Podger and Deon Canyon

Published 2007-10-31

For quite a while we have been designing and writing a small, single-user application. It is an authoring environment for web-delivered case studies. There are no Clarion drop-down menus; what users see as the first window is a browse of their cases. It's a good way to drop them right into the driving seat.

The product, although small, is complex with quite a few drill-down pathways. Having decided against conventional Clarion menus, we began by populating the front window with good-looking buttons, one for each downwards pathway. The evil day eventually arrived: too many pathways, too many buttons and that top window, which had to look good, had become crowded and ugly.

We have a lot of third-party Clarion products, but we wondered if we could write a sidebar menu ourselves. Thoughts like these beguile the mind.

The temptation was too great, and this article is proof. It describes a simple home-grown menu, provides the source and delivers a single-window demonstration app. But a retail quality accordion menu it is not. The source is quite straight-forward and the places where it is modified are also clear, but some hand coding is needed to adapt it to particular needs. We use it ourselves, so it has had a workout. Want to improve it, make it shorter or turn it into a template? Feel free.

Appearances

When users run the app they see one of the two menus shown in Figure 1. The first of the pair has normal buttons, while the second has buttons created by **MakeOver**, a CapeSoft product. As explained above, in the real-life app there is also a browse and the menu is alongside it, but here the focus is just on the menu, so the demonstration application shows only that.



Figure 1. The menu on entry to the app (with and without Makeover).

From now on all figures will show MakeOver buttons, but the provided source has only the standard ones. It is a simple matter to make these 'flat', if so desired.

As well as being buttons, each menu entry is also a Region. That is, the MouseIn event is detected and acted on before any mouse-click event can occur. Appearances are everything in this case.

The downward-arrow icon on each menu entry indicates that it expands to show a sub-menu. When the mouse is hovered over the top entry, it expands as shown in Figure 2.

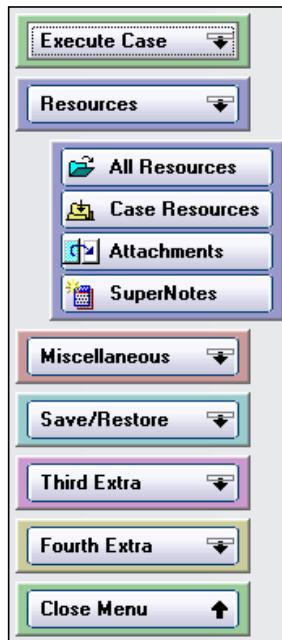


Figure 2. First sub-menu expanded

In Figure 2, please note that the menu entries below the top entry have moved downwards to accommodate the unhiding of the sub-menu. The general principle is that only one sub-menu is visible at a time. In fact, the sub-menu shown in Figure 2 is only ever displayed in this unchanging X,Y position. It is hidden and unhidden at this position and does not move. Rather it is the top-level menu entries that have to move up and down to make way for it.

What is true for the first sub-menu is true for all of them. They appear and disappear in their designated positions and move neither up nor down. How do they look in the IDE?

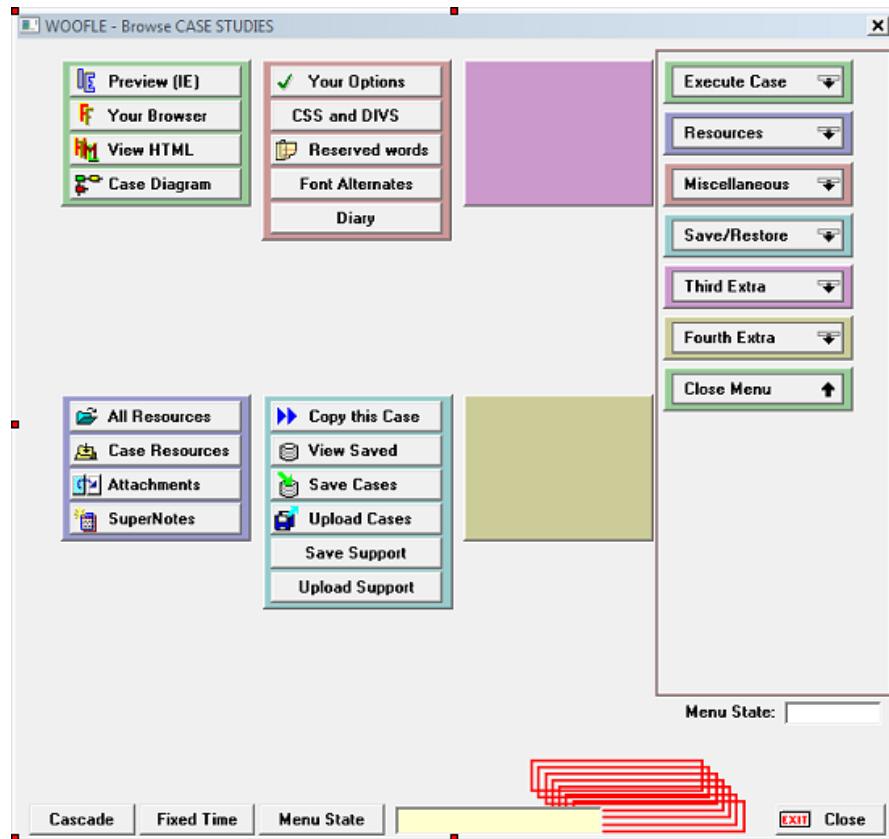


Figure 3. The window layout in the demonstration program, accord.app ([view full size image](#))

Figure 3 shows the sample application's menu in the window formatter. In this case the submenu items are spread out for clarity and moved into place at runtime with SetPosition statements. In our working application, however, we tolerate all the menu and submenu controls stacked up one on top of the other (Figure 4). It is an unpleasant sight, and just looking at it alongside Figure 3 forces us to contemplate fixing the original. But the point is that programming an accordion menu is *not as hard as it looks*. Sub-menus only display in one position.



Figure 4. The window layout in our production app

Bear in mind that it is the 'stay put' content that gets bigger or smaller as menu entries are added or deleted. To put it in a nutshell, the components with a *fixed* content (the top-level entries) must move but the components with *variable* content (the sub-menus) do not. This is what makes for programming simplicity, as will be seen when the source code is presented and explained.

In the demonstration program, accord.app, there is just one window, as shown in Figure 3. It has been lifted out of the production program, had its browse removed and its sub-menus placed where they do not obscure each other. There is plenty of room for each of the sub-menus to grow to their full size of eight entries each.

There are also some red boxes in the lower right corner which can be used to trace the movement of Regions during debugging. By moving a box to the same location as its Region and unhiding it, errors in Region placement are easily seen.

The demonstration program allows six top-level menus, with a seventh, Close Menu, being there just to close up and hide any open sub-menu. Each sub-menu may have up to eight entries. These limits are soft and can be extended by enhancing the source.

From a user's point of view, moving the mouse up and down the menu has the effect of opening one sub-menu at a time and closing the others. The top-level menu entries relocate to accommodate the appearance of each variable length sub-menu and the disappearance of the others. Clicking on a sub-menu entry executes the code linked to its button. On return to the menu, after this execution, the menu is exactly as it was before the linked procedure ran. The menu's behaviour is stable and fast.

Having established the appearances, it is time to get under the hood.

Menu sizes and placements

The menu's position is configurable. Firstly, some variables are needed, as shown in Figure 5.

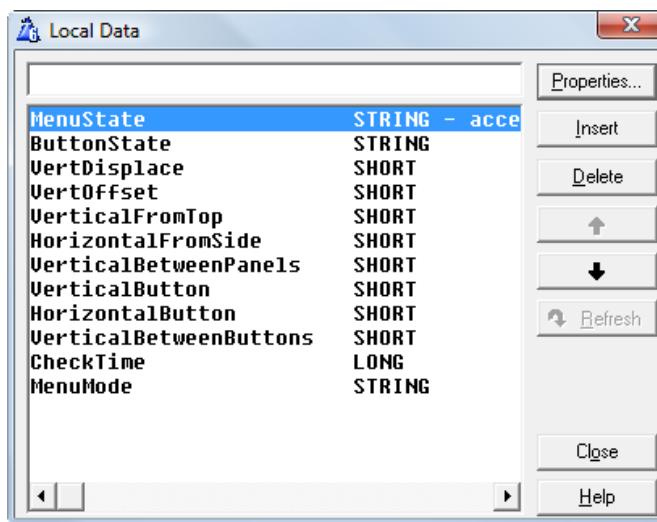


Figure 5. Required local variables

The solution we are describing is taken directly from our app and, as a result, is not abstracted. The names used are specific to the app. This is probably a help for those who best understand how things work in practical, applied terms. Those wanting a more generalized solution can change the names to suit. With that kind-of apology out of the way, here is an explanation of each variable:

ButtonState

The names of the different button states are Execute, Resources, Miscellaneous, Save/Restore, Extra, Fourth and Concealed. Taking the last one first, in the Concealed state only top-menu entries are shown, all sub-menus are *concealed*.

The other six states indicate which of the sub-menus is visible, but the last two are spare and in the example display only an empty panel.

Six variables and one array's values are set by the programmer with simple assignment statements. The assigned values control the placement of the menu within a window and set various other dimensions, such as the placement of panels and buttons in relation to each other.

- VerticalFromTop - Vertical distance from the top of the window to the top of the first panel.
- VerticalBetweenPanels - Vertical distance from the top of the first panel to the top of the next panel. Applied as the vertical distance between a top-level panel and its neighbour.
- HorizontalFromSide - Distance from the left side of the window to the left side of a top-level panel.
- VerticalButton - Vertical distance between the top of a panel and the top of its nearest button.
- HorizontalButton - Horizontal distance between the left side of a panel and the left side of its nearest button.
- VerticalBetweenButtons - Vertical distance between the top of a button in a sub-menu and its nearest button.
- VertOffset - An array of vertical offsets, one per sub-menu, specifying the vertical space needed by each sub-menu.

These values are used as well to derive the height of each sub-menu panel.

Some array values and one variable are calculated:

- VertDisplace - An array with one value per sub-menu. The calculations are described later when the source code is discussed.
- CheckTime - Saves the time to protect against accidental menu cascading. Usage described when source is discussed.

It should be noted that the above values over-ride the visual placements shown in Figure 3.

The code

On opening the window the following code is executed. It is at this embed point the programmer can modify the soft placements and sizes:

```

CheckTime = 0      ! initialize time variable
ButtonState = 'Concealed' ! open with all sub-menus concealed
VerticalFromTop = 9    ! distance of top panel from top of window
HorizontalFromSide = 293 ! and from the left of the window
VerticalBetweenPanels = 24 ! vert. distance: PanelN to PanelN+1
VerticalButton = 3      ! vert. distance: PanelYPos to ButtonYPos
HorizontalButton = 4      ! horiz. distance: PanelXPos to ButtonXPos
VerticalBetweenButtons = 16 ! vert. distance between sub-menu buttons
DO InitializePanelPositions
DO ShowButtonState

```

The InitializePanelPositions routine, as explained above, does a once-only relocation of all sub-menus to the positions they will occupy throughout execution. This code can be found in accord.app, available for download with this article.

The ShowButtonState routine contains the greater part of the code in the demonstration app. It is run, as shown here, when the window opens, but is mostly invoked when the mouse cursor enters a Region. It will help to discuss this usage first before getting into the details of the routine. Figure 6 shows an example. If a Region is made Immediate, that is, it is set to detect events as they occur, the MouseIn embed point will be shown, otherwise it will not.

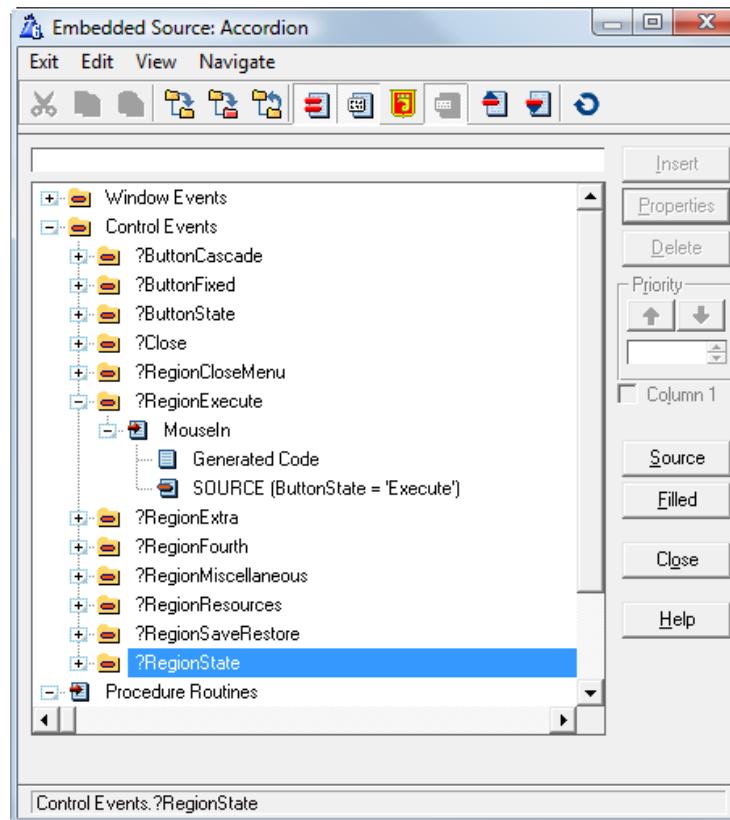


Figure 6. The MouseIn event embed

The entire embedded code for the MouseIn event shown in Figure 6 is only two lines:

```
ButtonState = 'Execute'  
DO ShowButtonState
```

The effect of this code is to trigger the ShowButtonState routine whenever the mouse cursor enters the enabled Execute Region. The routine picks up the supplied ButtonState value and acts on it. This routine is used for every Region. The only difference is the value written to ButtonState before the routine is called.

Here's the routine code in detail.

At the very beginning of the ShowButtonState routine you'll see the following code:

```
Time# = CLOCK()      ! get the time right now  
IF CheckTime        ! if there is a previous time  
  Difference# = Time# - CheckTime ! find the difference  
  IF Difference# < 33    ! ignore mouse events that happen too soon  
    EXIT  
  END  
END  
CheckTime = Time#    ! save the time of allowed events
```

Before we added this code a curious bug troubled us. When a new sub-menu appeared, it pushed down the top-level menu entries below it and, with them, their associated Regions. It finally dawned on us that any old Region could now have been pushed under the mouse cursor, triggering an unwanted MouseIn event and activating another execution of the ShowButtonState routine. *This* execution could slide yet another Region under the cursor, causing an inexplicable

cascade of events. Sub-menus would snap open and shut so fast that it was impossible to see what was happening. This is why there are small red boxes in the lower right of the window displayed in Figure 4. Using them, the above events can be traced.

The effect of the above code is simply to ignore a MouseIn event that occurs within a third of a second of the one before it. Humans will normally take longer between their menu choices, so this code ignores the events that make up the unwanted cascade. A very quick user will just have occasionally to mouse away from a menu entry and mouse back in again.

The next code to be executed in this routine is:

```
VertOffset[1] = 72    ! these are all the offsets
VertOffset[2] = 72    ! this is the only place you change them when
                      ! any sub-menu gets more or less buttons
VertOffset[3] = 88
VertOffset[4] = 104
VertOffset[5] = 56
VertOffset[6] = 40
VertOffset[7] = 72
```

A sub-menu panel containing two buttons, each 16 DLUs high, has a height of 40 owing to a padding of 4 top and bottom ($16+16+4+4 = 40$). Adding another button increases the height to 56, while a fourth button takes the height to 72, and so on.

The rest of the ShowButtonState routine is devoted to dealing with each invoked ButtonState.

The source code for the processing of the Execute state will now be examined a bit at a time. It falls under a Case statement:

```
CASE ButtonState
```

The first part of the code reads as follows:

```
OF 'Execute'
DO UnHideExecutePanel
DO HideResourcesPanel
DO HideMiscellaneousPanel
DO HideSaveRestorePanel
DO HideExtraPanel
DO HideFourthPanel
```

This code hides all sub-menus except the Execute sub-menu, which it unhides. The code is very simple. As an example, consider the HideResourcesPanel routine:

```
HideResourcesPanel ROUTINE
HIDE(?PanelRealResources)
HIDE(?ButtonAllResources)
HIDE(?ButtonAllResources+1000)
HIDE(?ButtonCaseResources)
HIDE(?ButtonCaseResources+1000)
HIDE(?ButtonAttach)
HIDE(?ButtonAttach+1000)
HIDE(?ButtonNotes)
HIDE(?ButtonNotes+1000)
```

The only unusual thing about this brief routine is the expressions with a +1000 in them. This is MakeOver at work. Its buttons have a background that must be hidden along with the button itself. Arbitrarily, the

```
HIDE(?ButtonAttach)
```

command must be accompanied by

```
HIDE(?ButtonAttach+1000)
```

or the background will stay on screen. All the Hide/Unhide routines can be read in the provided example app.

The next part of the code is more interesting. It reads:

```
LOOP D#= 1 TO 8
    VertDisplace[D#] = VerticalFromTop ! initialise with vertical
        ! distance into window
    ! now establish the rest position displacements
    VertDisplace[D#] += (D# - 1) * VerticalBetweenPanels
    IF D# > 1
        ! now add extra displacement caused by revealing sub-menu panel
        VertDisplace[D#] += VertOffset[1]
    END
END
DO ApplyVerticalDisplacements
```

The above code builds the content of the VertDisplace array. Each member of the array is first initialized with the VerticalFromTop value. A multiple of the VerticalBetweenPanels is then added in, using the loop value (D# - 1) as the multiplier. Lastly, another value is added that takes account of the height of the unhidden sub-menu panel. The ApplyVerticalDisplacements routine is then executed. Its full contents can be read in the provided example app. A sample of this code is sufficient to explain it:

```
?PanelExecute{PROP:YPos} = VertDisplace[1]
?RegionExecute{PROP:YPos} = VertDisplace[1]+VerticalButton
SETPOSITION(?ButtonExecute,,VertDisplace[1]+VerticalButton)
SETPOSITION(?ButtonExecute+1000,,VertDisplace[1]+VerticalButton)
?PanelResources{PROP:YPos} = VertDisplace[2]
?RegionResources{PROP:YPos} = VertDisplace[2]+VerticalButton
SETPOSITION(?ButtonResources,,VertDisplace[2]+VerticalButton)
SETPOSITION(?ButtonResources+1000,,VertDisplace[2]+VerticalButton)
```

Briefly, the top-menu panel is moved vertically to the Y position given by the calculated VertDisplace value. The Region follows, being displaced a little further to allow for the VerticalButton value. Finally, the button itself and its MakeOver ghost are also moved. If the PROP:Ypos method is used to move the ghost a compile error is thrown, hence the use of SETPOSITION.

Lastly, some tidying up. The buttons on the sub-menu panel are unhidden and the CloseMenu is attended to. The code is:

```
UNHIDE(?ButtonExecute)
UNHIDE(?ButtonExecute+1000)
UNHIDE(?ButtonResources)
UNHIDE(?ButtonResources+1000)
```

```

UNHIDE(?ButtonMiscellaneous)
UNHIDE(?ButtonMiscellaneous+1000)
UNHIDE(?ButtonSaveRestore)
UNHIDE(?ButtonSaveRestore+1000)
UNHIDE(?ButtonExtra)
UNHIDE(?ButtonExtra+1000)
UNHIDE(?ButtonFourth)
UNHIDE(?ButtonFourth+1000)
UNHIDE(?PanelCloseMenu)
UNHIDE(?ButtonCloseMenu)
UNHIDE(?ButtonCloseMenu+1000)
ENABLE(?RegionCloseMenu)

```

So, there it is, a low-budget but useful accordion menu. But there was a niggling discontent: sometimes the menu felt sluggish.

There is a bit too much compromise here, the trading away of a third of a second to avoid the unwanted event cascade mentioned above. There is another solution, however, and it is interesting in that it introduces a new, larger Region that overlaps all the rest.

To explain, consider that the menu as a whole needs either to accept the mouse hovering on its menu buttons or ignore the mouse. Its beginning state is to accept, but, as soon as a hover over any top-level menu entry triggers the first execution of the ShowButtonState routine, then the menu state flips to Ignore. The problem then is to have the menu state revert back to Accept in time for the next valid hover. Is there some event that would recognize the need for a state change immediately?

Well, there is one thing that incontrovertibly must happen to bring about the next valid hover and that is the person holding the mouse has to move it! And to detect this movement there must be a new Region encompassing the whole menu. We also add a new variable, MenuState, a STRING to hold the Accept or Ignore state.

The code for the MouseIn events (see Figure 6) gets another line, which changes the menu to the Ignore state:

```

ButtonState = 'Execute'
DO ShowButtonState
MenuState = 'ignore'

```

To understand how MenuState is returned to its Accept state, study Figure 7, which shows the large Region (?RegionState) needed to detect a MouseMove event. To make it visible, a box has been placed in the window of the same size and in the same location as ?RegionState. This Region encompasses all the top-level menu buttons.

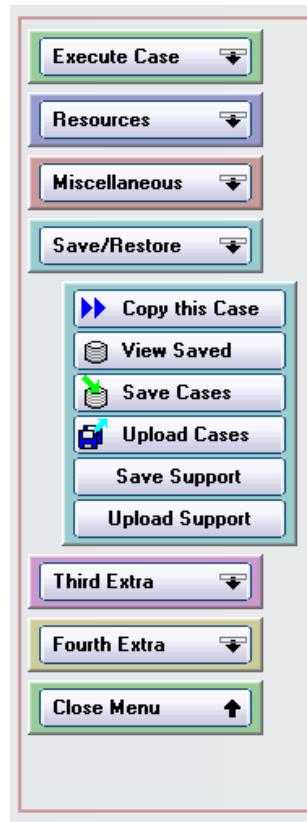


Figure 7. Size and Location of ?RegionState

The sequence of events to display the submenu shown in Figure 7 is as follows:

1. A MouseIn event has been detected by ?RegionSaveRestore.
2. The routine ShowButtonState is executed, making the sub-menu visible.
3. The menu's state becomes Ignore and stays this way until the mouse is moved.
4. Movement of the mouse within ?RegionState is detected by the MouseMove event connected to that Region.
5. The menu's state reverts to Accept, ready to detect the next MouseIn event.

It is important to note that ?RegionState is defined *after* the menu buttons. If this is not done the MouseIn events connected to these smaller buttons will not be detected.

The Property List for the window, shown in Figure 8, indicates the correct placement for ?RegionState.

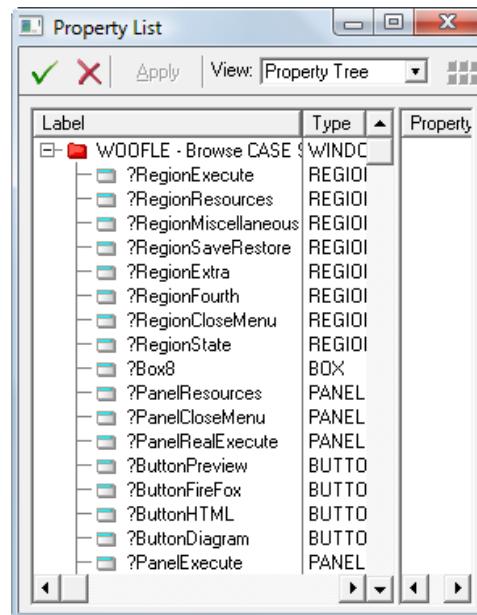


Figure 8. Placement of ?RegionState

The embed point under Control Events is shown in Figure 9.

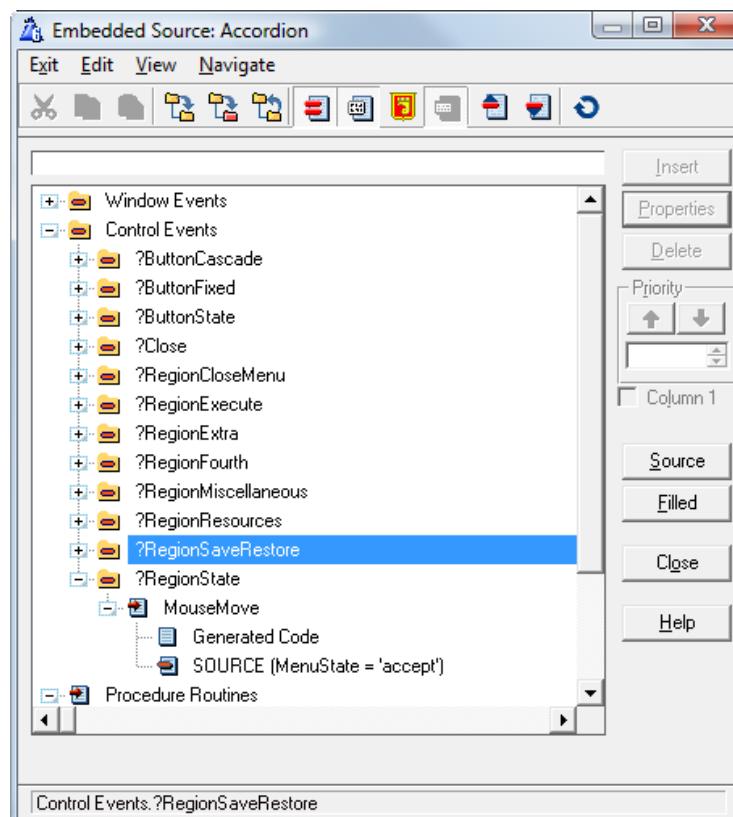


Figure 9. Detecting the MouseMove event

The entire code is the one line shown in Figure 9:

```
MenuState = 'accept'
```

When this state is examined at the beginning of ShowButtonState, it permits the MouseIn event to be processed.

Calling procedures

Calling code from the menu is really easy - after all, these are normal button controls. Just use the usual options available for buttons, and remember to only assign procedure calls to submenu items.

Summary

Our unique application design requirements led us away from a conventional Clarion menu to a list of buttons, and when that became too complex, we arrived at the nested button menu described here. The code is more straightforward than you might think, and the menu is easy to use.

[Download the source](#)

David Podger is from the old school. He joined IBM in Sydney, Australia in 1961 and has a Ph.D. from the 70's on the theory of software design, which he wrote in Manchester, England. These days he is focused on distance education and on the delivery of interactive case studies over the web.

Deon Canyon is an academic attached to the James Cook University in northern Australia. He specializes in distance education in the field of Public Health and develops interactive case studies about disease outbreaks.

Reader Comments

[Add a comment](#)

Clarion Magazine

Passing Variables To A Thread

by Maarten Veenstra

Published 2007-10-30

In one of my applications I have a procedure that generates a Word document (using Capesoft's *Office Inside*). This procedure is called from several places in my application; in each case it opens an (hidden) instance of Word, does its magic, prints and saves the document, and closes Word.

My customers asked to be able to batch-generate and print Word documents, so I wrote a process that calls this procedure repeatedly. But every time a new instance of Word is created and destroyed, which is silly because the entire batch can be done using the same instance. Apart from this logic, one customer, running Citrix, complains that the Word instance is not destroyed, so the batch is filling up memory with instances of Word.

In either case the solution is embedding the "Word generation" code in the batch process or letting this procedure run on its own thread, waiting for a new job. I choose the latter.

But then I discovered a new problem. This procedure accepts six parameters to enable it to do its magic. When you want to "talk" to another thread you use NOTIFY(), but with NOTIFY() you can only pass one parameter. After asking questions in the newsgroups and snooping around in the online help I found out that one parameter is enough; it can be used to pass an address of a class which contains the parameters as properties.

Since my Word-generation procedure is already called from several places in my multi-DLL app, I did not want to change how the procedure is called. So I renamed the original procedure and created a stub (source) procedure with the original name and parameter list, in the same DLL. I modified the original procedure so it is now STARTed from the application Frame on program startup where I hold its thread number in the global variable GLO:WordThread. After being STARTed this procedure no longer starts generating a Word document when called; instead it sits in a loop waiting to be notified of a pending job. The stub procedure with the original parameter list simply notifies this procedure of a single job, then waits for a return notification that the job is complete.

Setting up the data

Under the global data button I entered a class TYPE, as show in Figure 1.

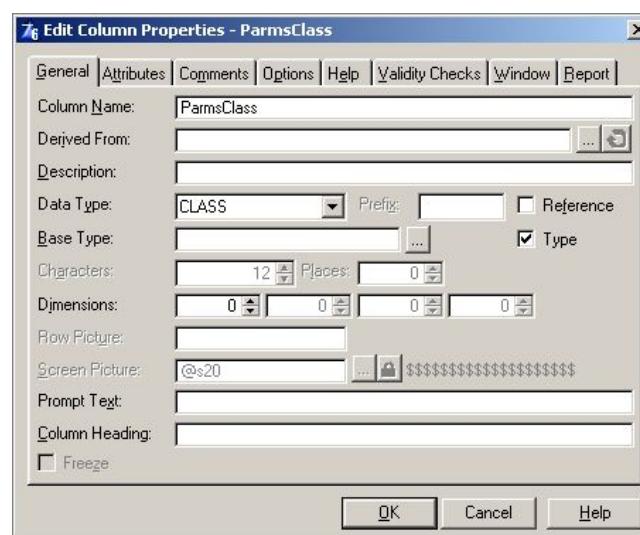


Figure 1. Creating the TYPED ParmClass declaration

And entered the parameters to pass as data members (properties) as shown in Figure 2.

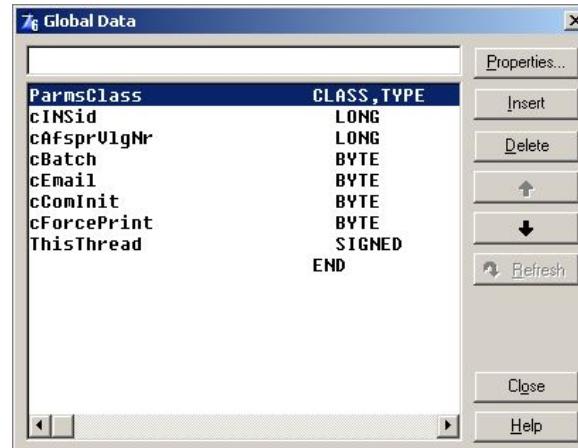


Figure 2. Entering the class's data members

I also entered some equates, in global data, to make the program more readable:

```
NOTIFY:Make    EQUATE(10)
NOTIFY:Close   EQUATE(11)
NOTIFY:Done    EQUATE(12)
NOTIFY:Failed  EQUATE(13)
```

The stub procedure has the original parameter list: (Long pINsId, Long pAfsprVlgNr, Byte pBatch=0, Byte pEmail=0, Byte pComInit=0, Byte pForcePrint=0). Since the calling procedures will wait until the "Generate Word Doc" procedure completes its task, the stub procedure has to wait as well. I used a general window template to create the stub so I had an ACCEPT loop in this procedure. Under the stub procedure's data button I instantiated the ParmClass by entering the name of the global typed class as the Base Type, as shown in Figure 3.

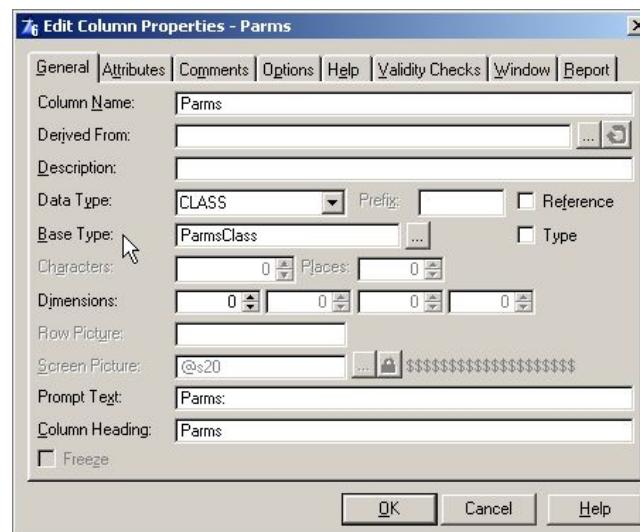


Figure 3. Creating an instance of the ParmClass type

Directly after the CODE statement I assign the passed parameters to this class's properties:

```
Parms.cINsId    = pINsId
Parms.cAfsprVlgNr = pAfsprVlgNr
Parms.cBatch    = pBatch
Parms.cEmail    = pEmail
Parms.cComInit  = pComInit
Parms.cForcePrint = pForcePrint
Parms.ThisThread = THREAD()
```

And, in the EVENT:OpenWindow embed I NOTIFY() the Word thread to generate my document:

```
NOTIFY(NOTIFY:Make, GLO:WordThread, ADDRESS(Parms))
```

The first parameter of NOTIFY() defines the requested action. The second parameter is the thread number to receive the notification, and the third parameter is the one "free" parameter you're allowed to pass. In my case I use it to send the address of my Parmsclass. After that, this procedure remains idle until it gets a message back, which I capture in EVENT:Notify:

```
IF NOTIFICATION (NCode)
  ! IF Ncode = NOTIFY:Done
    POST(Event:CloseWindow)
  ! END
END
```

In the original Word generation procedure I added a reference to the global typed class, under its Data button, as shown in figure 4.

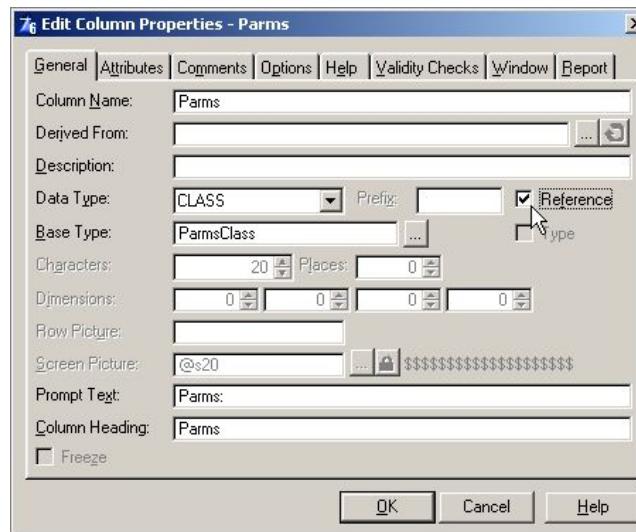


Figure 4. Adding a reference to of type Parmsclass

This needs to be a reference since I don't want to instantiate a class in this procedure but I want to refer to the instance of the class, created in my stub procedure. This procedure will come into action when it is asked, via NOTIFY(), to perform a task.

Under EVENT:Notify I've embedded this code:

```
IF NOTIFICATION (NCode, Nthread, NParam)
CASE Ncode      !Test the Notify Code
OF NOTIFY:Make
  !Assign passed parameter to reference var
 Parms &= NParam + 0
  PAS:INSid  = Parmsclass.cINSid
  PAS:AfsprVlgNr = Parmsclass.cAfsprVlgNr
  BatchRun   = Parmsclass.cBatch
  PAS:Email   = Parmsclass.cEmail
  pComInit   = Parmsclass.cComInit
  pForcePrint = Parmsclass.cForcePrint
  CallThread  = Parmsclass.ThisThread
  LOC:Action = 1
  QuickWindow{PROP:Timer} = 1
```

```

OF NOTIFY:Close
POST(EVENT:CloseWindow)
END !case
END !if

```

The first parameter of NOTIFICATION() defines the requested action. The second parameter is the thread number of the sender and the third parameter is the free parameter.

As you can see, I reference-assign the third NOTIFY() parameter, which is the address of the class from the stub procedure, to my local class reference variable. The code

```
Parms &= NParam + 0
```

is a little trick to tell the compiler to assign the address, not the NParam variable, to Parms. You could also do it this way:

```
Parms &= (NParam)
```

That gives me access to all properties and methods of this class. In the original design of this procedure, the passed parameters were assigned to local variables. So all I had to do is assign the class properties to these same local variables and then start the timer loop which handles the generation of the Word document.

I'm checking for two notify codes here. Since this procedure runs forever, on its own thread, it cannot be closed in the usual way. The application frame sends a NOTIFY(NOTIFY:Close) from its Event:CloseWindow embed so this procedure can properly shutdown and kill its Word object.

When the Word generation is complete this procedure stops the timer loop and sends a NOTIFY() back to the caller thread to signal its completion. NOTIFY(NOTIFY:Done, CallThread) indicates the generation was successful, and NOTIFY(NOTIFY:Failed, CallThread) indicates the generation failed somehow (I omitted the failure code in the stub procedure for clarity).

Summary

Problems with calling a Word document generation procedure in batch mode led me to the solution of running the generation code on its own thread. But that raised another problem: how to pass in multiple parameters via NOTIFY. Passing in the address of a class proved to be the perfect solution, and the problem turned out to be very simple to solve, as is usually the case once you know how.

[Download the source](#)

Maarten Veenstra began his computing career as a field service engineer on the DEC PDP11 family and a microprocessor-controlled punchcard machine. His first personal computer job was fixing CP/M computers and the early IBM clones, during which time he bought an MSX "computer" and taught himself BASIC and assembler. Maarten began using Clarion Professional Developer in 1988, wrote his first Windows program with CW 2.0, and his first ABC program with C4. He is now the co-owner of [Nepucon](#), which provides service to public utilities. Maarten married in 1991, and he and his wife have adopted two beautiful Chinese girls.



Reader Comments

Posted on Wednesday, October 31, 2007 by Paul Howard

Is a CLASS really needed? Would a TYPEEd GROUP work?

Posted on Thursday, November 01, 2007 by M Veenstra

Paul,

Then I need to pass the address of the group as I cannot change the Notify() prototype to pass a group and then reference assign it on the receivers side in the same manner as is done now with a class.

Maarten

[Add a comment](#)

Clarion Magazine

Developer Bio: Ivan Grech Mintoff

Published 2007-10-19

Editor's note: Ivan Grech Mintoff is the new owner of Michael Brooks' ClarioNET thin client solution for Clarion developers. Although Developer Bios normally focus on the individual I've allowed Ivan some leeway in describing his plans for ClarioNET. Otherwise this interview follows the pattern of questions established by Sue Pichotta in the [IceTips Bios](#). And be sure to follow the links in the text to further information about Malta.

CM: I gather that you are a nephew of Dom Mintoff - I think a lot of my readers would be quite interested to hear a bit more about Malta, and perhaps an anecdote or two about your uncle.

Wow! How the heck did you find that out?! Yes he's my uncle and I'm very proud of him.

A couple of Christmases ago he gave me the very pen he used to sign the treaty through which the Brits left Malta once and for all (militarily) and from which the process of Malta becoming a republic started. He also used this pen throughout his time as Prime Minister. The pen was also used by Lord Carrington (British PM); in his autobiography Lord Carrington called Mr. Mintoff "the hardest negotiator that I have ever had to deal with"! The pen holds a special place in my heart, as it signifies the culmination of decades of struggle, all focused into one great historical moment which was achieved through peaceful means rather than any bloodshed.

There are other members of my family who, in my eyes, have reached equally remarkable goals through great personal sacrifice and utter conviction of their personal and religious beliefs.

Who do you work for?

I run two software companies: DGT Limited & FSM. One is for the local Maltese market and the other deals with the foreign market. Apart from Malta, I also have a business presence in Qatar and in Bahrain in the Middle East. This combination has given me a great opportunity to understand the differences in European and Middle Eastern cultures and it has truly been a pleasure to mix the two and see the best in both.

Working in the Middle East has also given me the opportunity to cut through news propaganda (on both sides), feel the every day life of both "blocks" and see how, in reality, we are very, very similar to each other.

Of course, now I have also taken over [ClarioNET](#) which an exciting new venture for me. It is taking up a huge chunk of my time as I intend to start on the right footing and make sure it reaches much higher levels than it presently enjoys.

What do you like best about what you do now?

The traveling, meeting new people, working with different types of organisations (sizes, culture, people) and on different types of projects, some very large and some small, struggling businesses.

The best part is seeing the look on a client's face when a job is well done and knowing that you have somehow contributed to his business.

I also enjoy when we go up against very big companies who tell the client 'it cannot be done' and then you give them the solution to their spec....

CM: What has been one of your biggest challenges in using Clarion?

I think the biggest challenge is that when compared to MS etc, Clarion is simply not a known entity to most organisations. When you mention what you will be providing and what programming language will be used, you automatically have a struggle on your hand with managers (especially non IT ones!) who live by the "no one ever got fired by using Microsoft" rule...

CM: Do you use any computer languages besides Clarion?

Although I have used other languages in the past, from Pascal, Cobol, assembly, VB etc., I now only programme in Clarion. Clarion covers my needs for my type of work.

I'm not interested in internet browser based technology, for which there are plenty of tools. I'm more interested in business software that in the majority still works on a Windows platform.

ClarioNET on the other hand is a completely different tool to Web Browser templates.

Clarion:

- Is simple to implement, you just pop it in and compile.
- Doesn't require knowledge of other languages for embeds etc.
- Is very fast.
- Gives the users the same layout as when working in windows (no HTML approximation).
- Is real thin client technology.
- Has built-in dynamic key Blowfish encryption.
- Works with a whole range of other templates (auditing, logging, reporting etc) that make the final app very polished.
- Is great for a business whose IT guys worry about updating clients, security etc.

The list is endless....

Ivan and Carolanne



CM: What's the coolest project(s) you've worked on using Clarion?

There are too many for one to stick out above the rest. Here are a few:

1. An order for 18,000 CDs with the Telephone Directory for Maltacom PLC. They distributed the standard CD for free to their users who could then upgrade to the enterprise version directly from us. Enterprise included a word processor, label/envelope printing, emailing etc. so would be bought by companies who wanted to use the data supplied and not just look it up.
2. Five years ago, I wrote an app for Malta Financial Services Authority. Basically, they register and monitor (via onsite visits) all banking, financial and insurance companies in Malta, licensing them according to categories. The app handles initial application, annual returns, fines, write ups, follow ups, license revocation, tracking of correspondence, fines, key staff, dates, automated emailing etc. It uses ClarioNET and of course there is an audit trail, backup/recovery etc. I wrote it five years ago, went live and have not had a single call out since....

As I said, there are many other such projects that were a pleasure to be involved in.

CM: How did you come to be the new owner of ClarioNET?

When ClarioNET first came out, I remember contacting Michael and discussing having a dynamic key type of encryption on the new product. This is something which I believe leaves other encryption products (as well as other TCP/IP type transmission software) standing. Even a session using SSL will automatically drop if the key changes, which to me is a huge Achilles heel for SSL.

I still have the email correspondence between Michael and myself discussing this feature. Imagine my surprise when Michael implemented it in the next update!

Some time later, I was working on something else with one of Microsoft's Senior Consultant of Internet, Infrastructure, and Security in the Middle East who was simply knocked over by one of my ClarioNET applications, and we then went through implementing ClarioNET into an app via the templates. He even came to Malta twice to see it in action and he brought other colleagues to see it. I have an email from him where we discuss the technology as "an improvement of the current SSL implementation and perceived weak spots".

I truly believe that ClarioNET is a great tool which is seriously undervalued and under marketed.

You can therefore imagine that when Michael announced that he was considering selling, I was very interested. The rest, as they say, is now history....

CM: What are your plans for ClarioNET?

I think I first have to give you some background of the purchasing process. This will help to better explain the future plans.

As part of the process, we asked a fair chunk of the present end users four simply questions:

- a. Are you still using the product?

The majority replied that yes, they were still using it. Bear in mind that (surprisingly) there are literally many hundreds of users of ClarioNET.

- b. Do you intend to use ClarioNET in future releases of Clarion?

Again the majority said that they cannot afford *not* to, as their present clients need to carry on using this technology. Alternative technologies are either not secure enough or are too expensive.

- c. What present problems are you encountering and what do you want to see?

There have been some good recommendations. Michael and I have been through them. In theory, they are technically possible but we now have to implement them. Barring major hiccups, we'll get there!

- d. What are the present projects that you have used ClarioNET for?

Here, Michael and I were simply knocked over!! There are governments, banks, insurance companies.. Some projects are truly huge, as are their multi-national clients using the technology! Of course there are many, many smaller companies using this technology with equally impressive results.

With all this in mind the near future plans are:

1. Get the message over to present users that we are taking this product seriously and we are here to stay. I wish to give them what they really need right now which is:
 - o latest updates
 - o support
 - o enhancements
2. Market the product better. We already have a testimonials page on our site which explains what has already been achieved by programmers using ClarioNET and even names the end users (with their permission). This will be a two way process: to build up ClarioNET's image and also to promote the programmers who already have solutions that they can offer to new potential clients. There will be announcements on this front over the coming weeks.
3. Whilst we are steadyng the boat and building confidence in our users, I also wish to see new features in ClarioNET.

We are working on the present wish list that the end users have come up with and there have already been several releases with these updates being included. For instance, thanks to Michael Brooks & Friedrich Linder (amongst others), we recently fixed a problem which occurred in a particular circumstance when using Vista (we are finishing off the final testing on this issue).

I have near-future plans which are presently being actioned and other not-so-near-future ideas which I guess are too early to discuss but are very exciting nevertheless. At this stage, let's just keep our feet on the ground!

CM: Have you done anything for a living other than software development?

From tour guiding, to working for charities in the UK, reconstructing boats, working in a bank, being involved in a political party and other political entities... you name it!

CM: What are your hobbies/what do you like to do when you're not using Clarion?

Carol and I are lucky in that we have a 40 foot boat. Being right in the middle of the Mediterranean on a tiny island means

we Maltese are mad about the sea. We enjoy the social life that comes with the boat and the island, from cruising to other Mediterranean destinations to remaining local, barbequing, drinking lots of great wines.

In the last two summers, we even lived on board for about a month and I worked via laptop from the boat, even though home is about 20 minutes away!

CM: Married, children, grandchildren, other close family you want to mention?

Regrettably, we do not (as yet) have kids although we are in the process of being evaluated for adopting some children. So far, Carol has already conceded to having two so I guess I have a little bit more work to do to reach my target of five!!

CM: Where were you born?

I was born in Bulawayo, Zimbabwe (then Rhodesia). My father was a doctor in the Rhodesian army and so the first five years of my life were spent in Zimbabwe, Zambia, South Africa etc.

CM: Where do you live now?

I'm now based in Malta but for 23 years (from age 7 to 30) I lived and studied in the UK, initially as a boarder and then on to university etc. My parents would travel around the world (Australia, Asia, Europe) and my brother and I would then catch up with them over the holidays in some new exotic destination.

CM: What's interesting about where you live?

Although I have lived most of my life away from [Malta](#), this is a very, very [special place](#).

It is extremely safe. (I'd guess we've had maybe three murders in the past five years?) You wake up to sunshine and sea. Business is good. There are huge amounts of history and culture all around (we have temples which are two thousand years older than the pyramids).

For those who are into the crusades, Malta was the turning point for the Knights after the continual losses of the Holy Lands, Jerusalem, Rhodes etc.) Our history is peppered with Bronze Age, Roman, Arab, Norman, Carthaginian, French (Napoleonic), English rule. You name it they've been here and left their mark!

Today, we are an independent island with so much culture, and the entertainment is fantastic: great hotels, restaurants theatre, jazz festivals, nightclubs fireworks festival, festas every weekend in summer and of course beaches.

One of the highlights of summer is to go to the [three day jazz festival](#), on your boat, and hear the greats of jazz whilst you entertain guest with barbequed food and good beers/wines!

Hmmmmâ€“ perhaps we can persuade SV to have their next European Devcon over here??

CM: Have you lived any other interesting places?

All over the world (see above)

CM: Which person, from past or present, do you most admire and why?

My preferences are for people who believed utterly in something, went against the current and reached their goal.

Jesus has to be top of the list both as a person and in my spiritual belief. I am in what is called the [Neocatechumenal Way](#). This gives me a great basis on how to live my life in all aspects.

We're absolute nutters when it comes to Christianity - vbg! I like to think of us as modern day Templars, and they were pretty fanatical and dedicated!

I'm also proud of [my uncle](#) who has done much for our country, and other members of the family like Fr. Dionysius Mintoff ([Google the name](#) to find out more).

Others like Ghandi, Mandela, Pope John Paul II etc. are also heroes whom I admire immensely.



CM: What is your favorite food/drink?

Food: My mother's home cooking!! If out, local pizza or Indian.

Drink: Tea. I live on the stuff followed closely by wine, Jack Daniels & Coke and real Guinness in that order.

CM: What is your favorite type of music?

Neocatechumenal Psalms (sorry!). Followed by rock; my favourite group is Status Quo (that'll confuse a few!).

I also love most music, blues, classical etc. but definitely do not like house, garage etc.

CM: What is your favorite book?

Apart from the Bible? (There I go again. But it's true!) I go through phases of books: humour (Henry Wilt), biographies: Shaka Zulu, Long Walk to Freedom (Mandela) etc. For relaxation: Tom Clancy etc.

CM: Any favorite movies?

Clear and Present Danger, The Lion of the Desert, and Absolution (starring Richard Burton and Billy Connolly) since it was filmed at my old college and I took part in it along with my school mates at the age of around 13.

CM: If Clarion never existed, what do you think you would be doing at this time?

I think I would probably be a fisherman on a boat. I love the idea of being away from land, in the quiet, the throwing of the net not being guaranteed a catch - living entirely off God's providence!!

CM: Anything else you want to mention?

Although I have far too many things happening in my life at the moment, I am enjoying this part of my life very, very much. I am very much at peace with myself and living each day as it comes. Not living in yesterday nor tomorrow but today is what it is all about, for me. Yesterday is gone and no guarantee for tomorrow. I know too many people that plan for years ahead. And they miss the beauty of living today!

Thank you for the opportunity of this interview.

Reader Comments

Add a comment

Clarion Magazine

Exporting Sub-App Prototypes With A Utility Template

by Dragan Duric

Published 2007-10-18

There are at least two reasons why Clarion applications are split into main and sub-applications. One is to better enable team development, as described in Clarion Help (see the "How to" section "Creating a .DLL (Sub-Application)"). Another is to make it easier to reuse code in the form of a sub-application library.

To use sub-application or library procedures, prototypes must be defined in the main (or any calling) application within an external module. If the sub-application's prototypes change and those changes are not made in all calling applications, serious problems can occur.

In this article I'll review how sub-application procedures are declared in the main application, and I'll present a template to simplify the process of keeping procedure prototypes synchronized.

Adding procedure prototypes

You can add a single sub-application prototype to your main application by inserting a new module of the desired type, and then highlighting that module and choosing Procedure|New from the main menu. The procedure will be added to your application by a special template called External (ABC). The main function of this template is to register the procedure into application tree and to fill the %Procedure template symbol for prototype generation. The template does not write any source code (although the Application Generator will later create the appropriate code based on the procedure and module data). Here's the template source:

```
#PROCEDURE(External,'External Procedure','External'),HLP('~TPLProcExternal')
#AT(%CustomGlobalDeclarations)
#INSERT(%FileControlSetFlags)
#ENDAT
```

You can also create an external module (Application|Insert Module, then choose the module type) and use the module's Map Include File prompt. You'll need to specify a .clw or .inc file containing the external procedure prototypes.

The advantage of the external procedure template over the map include file is the ability to see the procedure and description within the main application's procedure tree.

The problem

If you have a sub-application with many procedures and parameters, it will take lot of repeated manual work to declare them all in your main application. And what if you make changes? The Clarion compiler will compile all procedure calls only if they match prototypes within the main application. But does the prototype in the main application exactly match the prototype in the sub-application? Everything depends on how much information the linker has about the exported procedure's parameter list.

Microsoft has laid down the following rule for DLLs (among others):

Procedures/functions with a fixed parameter list use the "stdcall" calling mechanism.

This is same convention as Clarion PASCAL procedure attribute (used for Win32API calls). In other words, if you're creating Windows-compatible DLLs you need to use the PASCAL convention.

The C compiler stdcall mechanism *decorates* procedure names. In particular, a procedure name like XXXX is translated to _XXX@nn where nn is the number of bytes of parameters passed to the procedure. The decorating scheme was established by Microsoft, and has been informally followed by other compilers including Digital Mars, Borland, and GNU gcc. Note that Windows does not decorate DLL API calls.

If you're not using the C or PASCAL attributes, Clarion will apply its own name decoration (name mangling), which will enable the linker to check not only for exported names but also to check if external prototypes match those in DLL, via the DLL's compiled LIB file.

Clarion 6 uses a non-standard but, in my opinion, much better approach. Clarion's name decoration is almost human-readable and includes all parameter types. A Clarion procedure declared as

```
subapp_proc2 PROCEDURE(byte, long, <long>, *long, <*long>, ulong)
```

will be generated into the sub-application's .exp file as

```
SUBAPP_PROC2@FUclOIPIU1
```

Clarion's name decoration is not described in documentation, and it seems to be different with different version of Clarion. The example procedure above is generated using Clarion 6; Clarion 5 decorates the function differently. It is easy guess that Uc stands for Unsigned character (byte), l stands for Clarion long, OI is Optional long, RI is probably Referenced long, PI is Pointer to long and UI is Unsigned long (ulong).

With SoftVelocity name decoration, the linker has enough information to compare the main application external prototype with the sub-application DLL (Lib). So why would you use anything else? In short, to create Windows-compatible DLLs, either because you're following a general policy of Windows compatibility, or because you want to call your Clarion DLLs from other non-Clarion applications.

Standardized DLLs

Under the C and PASCAL calling conventions, parameters are stored on the stack between procedures calls, but in opposite stack order. SoftVelocity languages pass procedure parameters in machine registers rather than using the stack. This generates smaller and faster code. In order to create a standard Windows DLL, however, Clarion sub-applications should export procedures using the PASCAL calling convention. This means that the exported procedure name is *not* decorated. If you link a Windows-standard Clarion DLL the compiler will not report an error on prototype mismatches. The same applies to functions using the C calling convention. Failure to correctly prototype your functions will almost certainly result in a General Protection Fault at run time, and the difference between parameters sizes in the main and sub-applications will probably result in stack corruption. Any difference in parameter position or size may cause unpredictable application behavior, and not give you any clue where to look for the bug.

My solution to this problem is to automate the exporting of prototypes from the DLL applications and import the prototypes into the calling applications, adding more reliability to main/sub-application development or anywhere you are using Clarion-generated DLLs. With synchronized prototypes, the compiler will tell you if anything's wrong in the

calling code (provided that you haven't done something like switched the order of two same-typed parameters).

The template

Having in mind the Clarion template language and how easy is to import application modules in the form of TXA text files, I decided to make a utility template that extracts procedure prototypes from a sub-application and generates a TXA containing an external module with the prototypes. I can then import that TXA module into the main application.

My first step was to export a sample external module from Clarion IDE into a TXA file and analyze it in order to recreate the necessary information from any sub-application. The second step was to write the template code, which I will now explain.

The template scans all procedures in application, selects exported procedures, and generates a TXA file. Well, that is not all. Sub-applications can be compiled as a DLL or LIB (library), using ABC or legacy templates, which gives at least four options that must be written into the template.

Furthermore, sub-applications can be exported in two ways. One is as an external module using generated procedures (see Figure 2, below). Another way is to generate an INCLUDED MAP file into a TXA generated external module, and not to generate procedures. Although I prefer a module with procedure prototypes, I have written the template with an option to generate a library with a Map Include File.

By way of demonstration I made a simple sub-application library (see Figure 1) and generated the following two TXA files.

The first TXA is for a LIB using generated procedures:

```
[MODULE]
NAME 'subapp.LIB'
NOPOPULATE
[COMMON]
DESCRIPTION 'subapp.txa'
FROM ABC ExternalLIB
[PROMPTS]
[PROCEDURE]
NAME subapp_proc1
PROTOTYPE '( long, long )'
NOEXPORT
[COMMON]
DESCRIPTION '( par_first, par_second )'
FROM ABC External
MODIFIED '2007/07/28' '12:39:35'
[PROMPTS]
[CALLS]
subapp_proc2
[PROCEDURE]
NAME subapp_proc2
PROTOTYPE '( long, long )'
NOEXPORT
[COMMON]
DESCRIPTION '( par_first, par_second )'
```

```

FROM ABC External
MODIFIED '2007/07/28' '11:57:38'
[PROMPTS]
[CALLS]
[END]
[END]

```

The second TXA is also for a LIB, but using a MAP include file:

```

[MODULE]
NAME 'subapp.LIB'
INCLUDE '.\subapp.clw'
NOPOPULATE
[COMMON]
DESCRIPTION 'subapp.txa'
FROM ABC ExternalLIB
[PROMPTS]
[END]
[END]

```

And here is the generated MAP include file ('.\subapp.clw'):

```

! For MODULE ( 'SUBAPP.ABC ExternalLIB' )
subapp_proc1      Procedure ( long, long )
subapp_proc2      Procedure ( long, long )
! END

```

Figure 1 shows the procedure tree for a sub-application to be synchronized to the main app

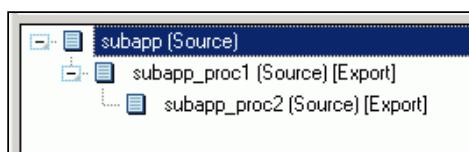


Figure 1. Imported library with procedures in main-application procedure tree

Figure 2 shows the main application with the imported external library module and the prototypes in generated procedures.

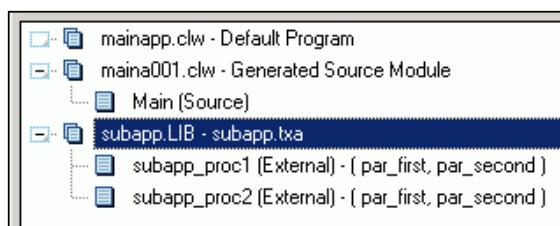


Figure 2. Imported module and procedures

Figure 3 shows the main application with the imported external library module and prototypes in an included map file. Figure 4 shows the module properties.

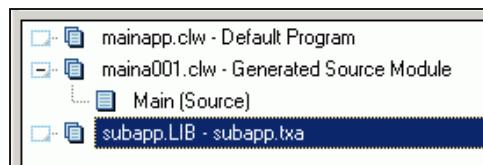


Figure 3. Imported module and included map.

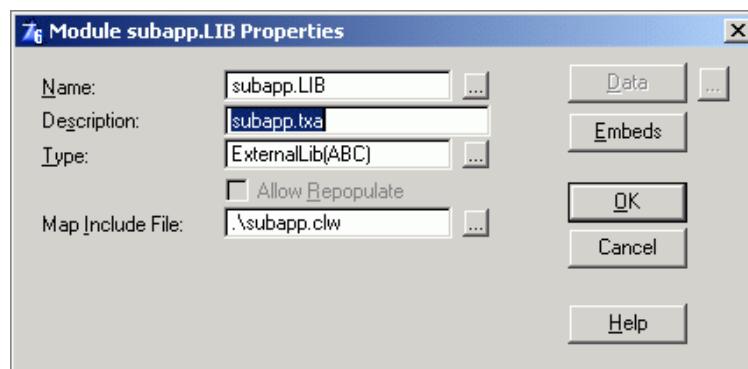


Figure 4. Imported external library module properties.

Using the template

Utility templates differ from other templates in the way they are activated. User have to start utility templates manually from the Clarion IDE, using the Application menu (Figure 5) or a hotkey (Ctrl+U). Of course, the sub-application must already be opened.

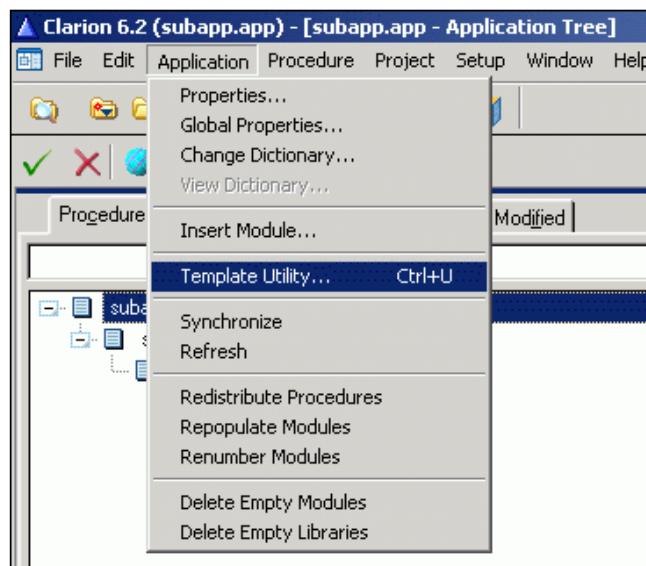


Figure 5. Running a utility template

Sometimes only some of the sub-application's exported procedures are required in the main application. This utility

template lets you select specific procedures. Other template options, as shown in Figure 6, include selecting based on included/excluded category, including procedures marked as exported (default), and including procedures which are external to the app.

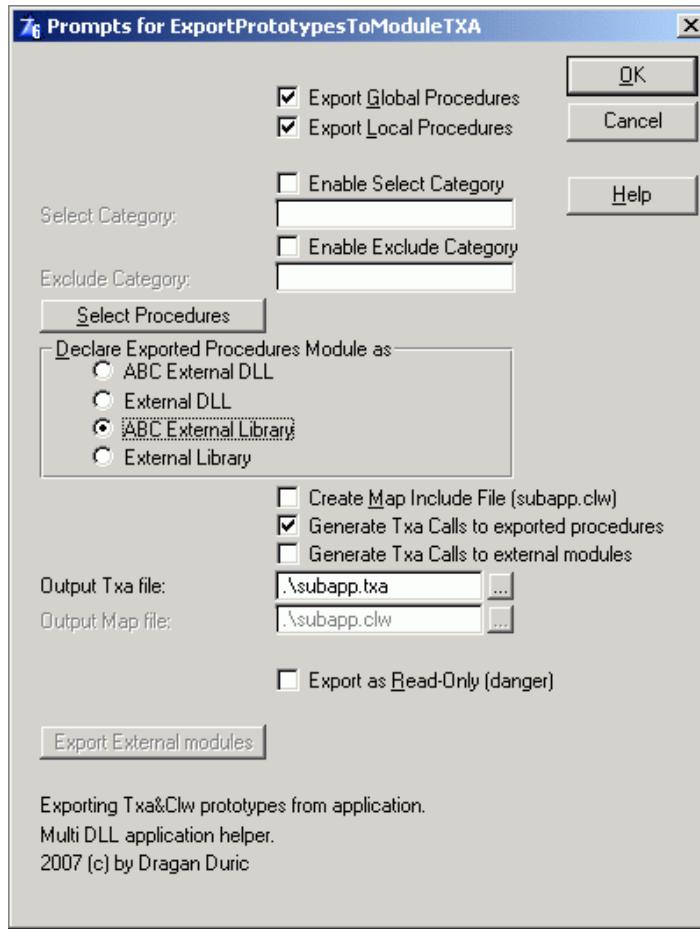


Figure 6. Export prototypes utility template prompts

After choosing the export options (do not forget to select the procedures to be exported), just press OK. Close the sub-application, open the main application (or any application that uses sub-application prototypes), chose File|Import Text..., and select the just-created TXA.

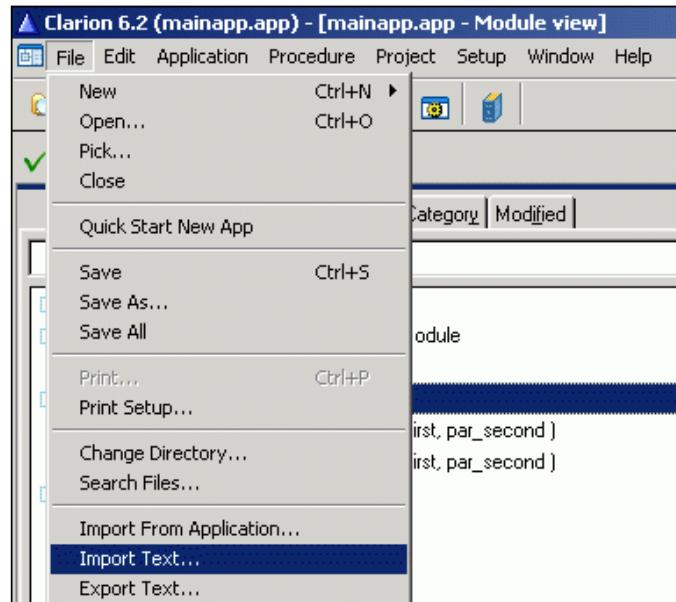


Figure 7. Importing Text from main-application

If you import a TXA created with the Map Include File option then you should first delete the obsolete module.

Unfortunately, importing a new version of an existing module will leave an empty module in the application tree. To remove redundant modules and libraries use the appropriate commands from the Application menu (see Figure 8). You can also delete the modules manually if you wish.

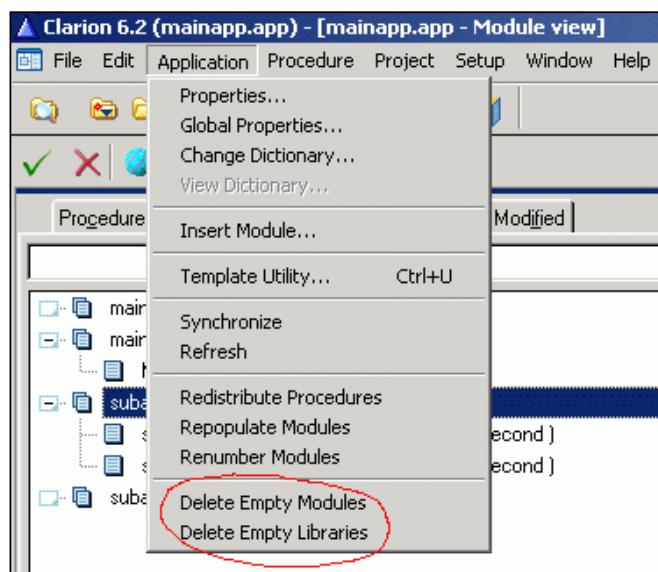


Figure 8. Delete empty Modules / Libraries

How it works

The template uses two prepared multivalued symbols. One is for the list of external modules (%ModulesExternal4Export), and the other is for the list of exported prototypes (%Procedures4Export).

Although the list of external modules is optional and used in the last #PROMPT, it is placed first, before any #PROMPT statements. The template uses a #PREPARE statement to load the list of modules before the template window

is displayed. The list of modules does not change while the template is in use.

```
#Prepare
#Declare ( %ModulesExternal4Export ), UNIQUE  #! list of external modules
#For ( %Module )
#If ( ~%ModuleExternal )
#Cycle
#EndIf
#Add ( %ModulesExternal4Export, %Module )
#EndFor
#EndPrepare
```

A second #PREPARE statement is placed after Select procedures #PROMPT; this code is executed every time you click on the Select procedures button.

```
#Prompt ( '&Select Procedures', FROM ( %Procedures4Export )), ↵
%ProceduresSelected4Export, SELECTION ( 'Select procedures' ), REQ
#Prepare
#Declare ( %Procedures4Export ), UNIQUE      #! list of procedures
#If ( %IncludeCategory )
#Set ( %ExcludeCategory, %False )
#Set ( %ExcludeCategoryName, " )
#ElsIf ( %ExcludeCategory )
#Set ( %IncludeCategory, %False )
#Set ( %IncludeCategoryName, " )
#EndIf
#For ( %Procedure )
#Fix ( %ModuleProcedure, %Procedure )
#If ( %ModuleExternal )
#Cycle
#ElsIf ( ~%ProcedureExported )
#Cycle
#ElsIf ( %IncludeCategory And %|
( ~instring ( upper ( clip ( %IncludeCategoryName ) ), %|
upper ( clip ( %ProcedureCategory ), 1, 1 ) ) ) #! Filter
#Cycle
#ElsIf ( %ExcludeCategory And %|
( instr ( upper ( clip ( %ExcludeCategoryName ) ), %|
upper ( clip ( %ProcedureCategory ), 1, 1 ) ) ) #! Filter
#Cycle
#EndIf
#If ( %ProcedureIsGlobal ) #! Filter
#If ( %ExportGlobalProcedures )
```

```

#Add ( %Procedures4Export, %Procedure )
#EndIf
#else
#If ( %ExportLocalProcedures )
#Add ( %Procedures4Export, %Procedure )
#EndIf
#EndIf
#EndFor
#EndPrepare

```

Create the MODULE file

After you press OK the template code after last #PROMPT or #DISPLAY statement executes. The #CREATE statement is used to create a new file or truncate an existing file to zero length.

```

#If ( %ExportTXAFileName )
#Create ( %ExportTXAFileName )

```

Generate the MODULE section

The module section is generated once.

```

[MODULE]
NAME '%Application.LIB'
#If ( %CreateClwMAPFileInclude )
INCLUDE '%CreateClwMAPFileIncludeName'
#EndIf
NOPOPULATE
[COMMON]
#If ( %ExportAsReadOnly )
READONLY
#EndIf
DESCRIPTION '%Application.txa'
FROM %ExportAsModuleTemplateType
[PROMPTS]

```

If Include MAP file is NOT generated

The following code executes only if you are *not* generating a map include file. The #FOR statement loops through each and every value of the multivalued % ProceduresSelected4Export symbol:

```

#If ( ~%CreateClwMAPFileInclude )

#for ( %ProceduresSelected4Export )

```

```
#Fix ( %Procedure, %ProceduresSelected4Export )
#Set ( %QuotedPrototype, Quote ( %Prototype ) )
```

In the Clarion IDE the prototype of an external procedure is visible only upon entry into procedure. To make prototype visible within application tree, I decided to fill in empty procedure descriptions with the procedure's %Prototype value. While exporting from Source Template (\$ProcedureTemplate = 'Source'), %Prototype is replaced with the %Parameters source template symbol. Here I discovered that %Parameters is not accessible without a #CONTEXT statement.

```
#If ( %ProcedureTemplate = 'Source' )
#Context ( %Procedure )
#Set ( %SourceTemplateParameters, %Parameters )
#EndContext
#Else
#Set ( %SourceTemplateParameters, " " )
#EndIf
```

Generate the PROCEDURE section

Here's the standard TXA [PROCEDURE] section:

```
[PROCEDURE]
NAME %Procedure
PROTOTYPE '%QuotedPrototype'
NOEXPORT
[COMMON]
#If ( %ExportAsReadOnly )
READONLY
#EndIf
```

Generate procedure DESCRIPTION

The DESCRIPTION line is a simple copy of the %ProcedureDescription. If the exported procedure has no description then the template uses the procedure prototype, and if the procedure template is Source then %Parameters is used as the description:

```
#If ( %ProcedureDescription )
DESCRIPTION "%ProcedureDescription"
#else
#If ( %SourceTemplateParameters )
DESCRIPTION "%SourceTemplateParameters"
#else
DESCRIPTION "%Prototype"
#EndIf
#EndIf
```

Generate LONG description

The long description is a text field of size 1000 (Clarion 6.2). As far as I know, it is not possible to simply insert the symbol into template code. Instead, %ProcedureLongDescription must be split into lines and generated on line by line basis. This code loops 16 times over a line size of 64 characters (1024):

```
#If ( %ProcedureLongDescription )
#Set ( %Counter, 0 )
#Loop, Times ( 16 )
#Set ( %LongDescr, %|
    Quote ( Slice ( %ProcedureLongDescription, %|
        ( %Counter * 64) + 1, %|
        ( %Counter + 1 ) * 64 ) ) )
#Set ( %Counter, %Counter + 1 )
#If ( %LongDescr )
    LONG '%LongDescr'
    #!Else
    #!Break
    #EndIf
    #EndLoop
#Else
    #If ( %ProcedureDescription )
        #If ( %SourceTemplateParameters )
            LONG '%SourceTemplateParameters'
        #Else
            LONG '%Prototype'
        #EndIf
        #EndIf
        #EndIf
    FROM %(LEFT (%ABC&' External'))
    MODIFIED '%(Format(%ProcedureDateChanged,@D10))' '%(Format(%ProcedureTimeChanged,@T4))'
    [PROMPTS]
```

Generate procedure CALLS

Here's the code to generate the procedure calls tree option (default true):

```
[CALLS]
#If ( %GenerateCalls )
#For ( %ProcedureCalled )
#If ( INLIST ( %ProcedureCalled, %ProceduresSelected4Export ) )
    %ProcedureCalled
#ElsIf ( %GenerateCallsExternal )
```

```

#Find ( %ModuleProcedure, %ProcedureCalled )
#If ( %ModuleExternal )
%ProcedureCalled
    #EndIf
    #EndIf
    #EndFor
#EndIf

#EndFor
#EndIf
[END]

```

Generate External MODULES

If the sub-application contains external modules (another sub-application and/or external DLL) and these modules are selected for export, the following code executes:

```

#For ( %ModulesExternalSelected4Export )
    #Fix ( %Module, %ModulesExternalSelected4Export )

[MODULE]
NAME '%Module'
#If ( %ModuleInclude )
INCLUDE '%ModuleInclude'
#EndIf
NOPOPULATE
[COMMON]
#If ( %ExportAsReadOnly )
READONLY
#EndIf
DESCRIPTION '%ModuleDescription'
#Set ( %TemplateClassAtStringPosition, %|
    InString ( '(', %ModuleTemplate, 1, 1 ) )
#Set ( %TemplateClassToStringPosition, %|
    InString ( ')', %ModuleTemplate, 1, 1 ) )
#If ( %TemplateClassAtStringPosition And %TemplateClassToStringPosition )
#Set ( %TemplateClass, %|
    Slice ( %ModuleTemplate, %|
        %TemplateClassAtStringPosition + 1, %|
        %TemplateClassToStringPosition - 1 ) )
#Set ( %Template, %|
    Slice ( %ModuleTemplate, 1, %|
        %TemplateClassAtStringPosition - 1 ) )
FROM %TemplateClass %Template

```

```

#Else
FROM %ModuleTemplate
#EndIf
[PROMPTS]
#For ( %ModuleProcedure )
#Fix ( %Procedure, %ModuleProcedure )
#Set ( %QuotedPrototype, Quote ( %Prototype ) )

[PROCEDURE]
NAME %Procedure
PROTOTYPE '%QuotedPrototype'
NOEXPORT
[COMMON]
#If ( %ExportAsReadOnly )
READONLY
#EndIf
#If ( %ProcedureDescription )
DESCRIPTION '%ProcedureDescription'
#Else
DESCRIPTION '%Prototype'
#EndIf
#If ( %ProcedureLongDescription )
#Set ( %Counter, 0 )
#Loop, Times ( 16 )
#Set ( %LongDescr, %|
      Quote ( Slice ( %ProcedureLongDescription, %|
                      ( %Counter * 64 ) + 1, %|
                      ( %Counter + 1 ) * 64 ) ) )
#Set ( %Counter, %Counter + 1 )
#If ( %LongDescr )
LONG '%LongDescr'
#Else
#Break
#EndIf
#EndLoop
#Else
#If ( %ProcedureDescription )
LONG '%Prototype'
#EndIf
#EndIf
#If ( %TemplateClass )
FROM %TemplateClass %ProcedureTemplate
#Else

```

```

FROM %ProcedureTemplate
#EndIf
MODIFIED %(Format(%ProcedureDateChanged,@D10))' %(Format(%ProcedureTimeChanged,@T4))'
[PROMPTS]
#EndFor
#EndFor
[END]
#Close ( %ExportTXAFileName )
#EndIf

```

Generate Include MAP file

If you are exporting prototypes as a library (LIB) TXA module, and if the Create MAP Include File checkbox is ON, the module procedures are *not* generated. Instead the module map include file is generated into the file defined by the TXA module INCLUDE prompt:

```
#If ( %CreateClwMAPFileInclude And %CreateClwMAPFileIncludeName )
```

Now create a new file using name of Map Include File:

```

#Create ( %CreateClwMAPFileIncludeName )
! Generated Map Include File
! For MODULE ( %(UPPER(%Application)).%ExportAsModuleTemplateType' )
#For ( %ProceduresSelected4Export )
#Fix ( %Procedure, %ProceduresSelected4Export )
%[33]Procedure Procedure %Prototype
#EndFor
! END
#Close ( %CreateClwMAPFileIncludeName )
#EndIf

```

Summary

Moving parts of applications into separate DLLs not only enables team development and reuse of procedure libraries, it will also shorten compile waiting times as individual applications are typically smaller. If your library prototypes are often changed (, or not), using this template will make your multi-DLL programming easier and less error-prone, especially if you use Windows-standard calling conventions.

I have released this template source code under the [Clarion Magazine Developers Open Source Public License \(DOSPL\)](#), so you are free to change it and use it in your commercial projects.

I wish you all happy programming.

[Download the source](#)

[Dragan Duric](#) is an independent software developer, addicted to Clarion since 1990. He has written accounting, POS and other administrative software in former Yugoslavia, Netherlands and United Kingdom.

Reader Comments

Posted on Thursday, October 18, 2007 by Stephen Ryan

Very Impressive!

[Add a comment](#)

Clarion Magazine

Tiers and Objects: The Future of Clarion.NET Development

by Dave Harms

Published 2007-10-11

I regularly hear from Clarion developers who really don't know what to make of Clarion.NET. Is the move to the .NET platform a good thing or a needless complication? How will Clarion programming change? Will those changes benefit my customers?

Until there's a Clarion.NET release many questions will go unanswered, but I believe it's possible to get a handle on the situation by looking at some of the general principles of .NET development. In previous articles on .NET I've explored the [.NET platform](#) and [.NET namespaces](#), and I've touched on some of the ways .NET applications [differ](#) from Win32 applications. In this article I'll explore how application architectures are changing, in the context of the move from Win32 to .NET. In doing so I'll try to answer the question of how Clarion programming is likely to change, both in the short term and the long term.

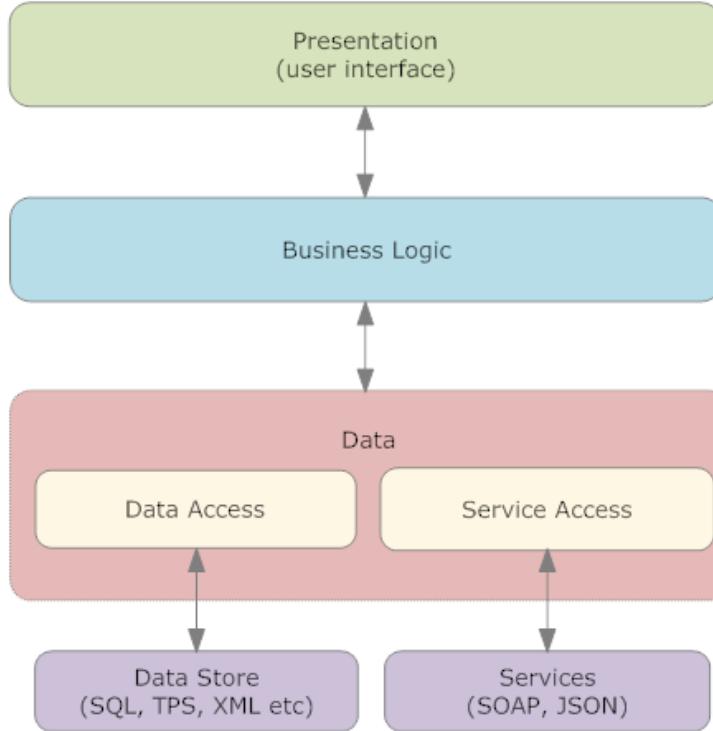
An application's architecture is the overall framework behind its design, and one of the very important changes in the move from Win32 to .NET is the sometimes gradual evolution of applications from single-tier to multi-tier.

How many tiers?

If you haven't already encountered the expressions *multi-tier* or *n-tier* in reference to applications you soon will. Actually multi-tier apps have been around for a very long time, and some Clarion developers may be creating multi-tier applications without realizing it.

But what is a multi-tier application anyway? To begin with, think of a very simple Clarion application using TPS files which are stored on the local drive. This application is a combination of a user interface, a data store, and the logic that allows the user to manipulate the data, and it's a single-tier application because there's no easy way to tease those aspects apart into separate, functional pieces. You can't, for instance, change the database from TPS to SQL on the fly. You can't, at runtime, swap out one piece of business logic for another piece of business logic.

By contrast, consider Figure 1, a diagram of a multi-tier configuration.

**Figure 1. A multi-tier application**

Multi-tier can mean many things in many different development contexts, but Figure 1 hits the high points for the kind of work most Clarion developers do now or are likely to tackle in the future. The key tiers, often called layers, include:

- Presentation layer
- Business layer (logic)
- Data layer (mediates access database or other data storage)
- Data store (the actual data) - this isn't a tier of the application itself, but is shown for clarity.

The presentation layer is the user interface, the various windows including menus, browses and forms. This layer handles the user's interaction with the system.

The business layer is the code that knows how to handle the data. If, for instance, this is an invoicing application, the business layer will contain the code for totalling amounts, applying taxes, and so forth. (Within the data layer you can see that I've shown the usual kinds of file-based data as well as data retrieved via web services.)

The important point about Figure 1 is that each layer only communicates with the adjacent layer; the user interface code doesn't know anything about the data store, and even the business layer is insulated from the actual database via the data layer. This last point is crucial; in this architecture your business logic will never contain actual file access statements such as GET/NEXT or SQL SELECT statements. Instead, the business logic layer talks to a data access layer, and that layer talks to the actual database, web service, or other data store.

The value of separating layers like this is in portability, ease of maintenance, and reusability.

You may have at some point faced porting a TPS application to SQL, or from one SQL server to another SQL server. The more tied your application's business logic is to the database's specific functionality, the greater the likelihood that code that works with one database will not work with another. If the business logic can deal with the data in a sufficiently generalized way, that logic becomes more portable.

Clarion developers have long benefited from this kind of abstraction thanks to Clarion's file driver system and, more recently, because of the further abstraction of the ABC library. If you really had to read/write dBase, TPS and SQL

files directly it would be a lot of work to move your application from one database to another.

The problem many Clarion developers encounter when porting to a new database is business logic that bypasses the built-in file access grammar. It's tempting to throw in some SQL SELECT or UPDATE statements to speed up a process, but there really is no SQL standard and a statement that works on one SQL server may not work on another. Now imagine that you have several places in your code where you need to execute the same kind of SQL statement. If you copy and paste that code, and something about the database changes, you have multiple changes to make, and multiple potential points of failure.

On the other hand, if you've isolated that code in a data access layer your business logic need never change, and presumably you'll only have one point of maintenance for each data access statement (SQL or otherwise).

And what applies to the business logic/data layer interface also applies to the business logic/presentation layer interface. Business logic that's tied to a particular interface is fine if you only have the one interface. But what happens when you need both a web version of your application and a desktop version. Are you going to duplicate the business logic? Or is the business logic available in a class or other library for both applications to use?

Distributed multi-tier applications

Things get yet more complicated when an application begins to spread itself across multiple machines. But really this isn't all that uncommon or unfamiliar. If you have a web application (built with one of SV's tools, [ClarioNET](#), or perhaps Capesso's [NetTalk](#)), and that application is getting its data from a dedicated SQL server, then you have an application that's distributed across at least three computers, and perhaps more if the server uses, say, a SOAP or other web service request to retrieve some data.

Multi-tier in Clarion (Win32)

I've suggested that multi-tier application design isn't utterly foreign to Clarion developers, but the reality is that in Clarion as we now know it this multi-tiersness, if I can use that word, is likely to be a messy combination of data tightly coupled to the user interface, mixed in with some specialized business logic classes (too often tightly coupled to the back end data), all propped up with a data access layer that only partly isolates the application from the back end data. Clearly this kind of development is missing out on most of the benefits of multi-tier design.

So where do we go from here? Is there a future for true multi-tier applications in Clarion Win32 development? Maybe, but I wouldn't expect to see much development along those lines. Clarion has long been a highly successful tool for client-server development, where the data is physically isolated from the application, but it would take a massive rewrite of the templates and the ABC library to accomplish any kind of multi-tier architecture. And many of the tools you'd need for that job aren't there, or aren't sufficient.

Tiers are not enough

Multi-tier architecture isn't just about tiers; it's also about objects, and particularly about two very useful features of modern object-oriented languages: reflection and introspection. These terms are often used interchangeably, although they have somewhat different meanings. Together they generally refer to the ability of a programming language/runtime environment to determine information about an object at runtime, and to the ability of a program to change its behavior based on that information.

What this means is that components have now become much more flexible, and communication between layers in an application is potentially much more a matter of "plug and play".

The Clarion language we know does have some introspective/reflective qualities. The WHO, WHAT, and WHERE statements let you determine information about data and objects and runtime, and you can use this capability to modify program flow. The .NET platform, however, provides introspective/reflective functionality far beyond what we can currently accomplish in Clarion. You can determine the type of a class at runtime, enumerate its methods, and

execute those methods as you choose. And Microsoft is continually expanding the runtime configurability, if you like, of .NET languages, with new ways of executing code based on the runtime evaluation of expressions. All of this makes for much more dynamic, component-oriented, flexible development.

Clearly, if Clarion is going to get to multi-tier development, it will be via Clarion.NET

Multi-tier development in Clarion.NET

So what is the multi-tier future of Clarion.NET application development? Here's what Bob Zaunere had to say in a [blog entry](#) about a year ago, responding to a question about the Clarion.NET templates:

The application architecture generated by the templates divides your application into layers: Presentation layer, Business logic layer (rules), and Data access layer. The templates allow you to make the decision on what type of data access layer you want to use. You could choose to have your data access layer generated using the familiar Clarion data access model. All of the Clarion data entities (FILE, RECORD, VIEW, QUEUE and GROUP) support data binding to any .Net control that provides for data binding. There are some obvious advantages to this model (and some not so obvious advantages). Or you can choose to have your data access layer generated to utilize the ADO.Net object model. We have DataAdapters for both FILE and VIEW structures that plug right into the ADO.Net object model, so you can happily work with what you know or the new ADO.Net objects. In the future you'll have the option to generate the data access layer utilizing an ORM model like nHibernate or Gentle.Net, or when its released using LINQ.

Leveraging Microsoft's .NET platform clearly makes it easier to get into the multi-tier game. So is Clarion.NET the Holy Grail of multi-tier development? Well, not quite, at least not off the hop.

The key here is the last sentence in the quote: "In the future you'll have the option to generate the data access layer utilizing an ORM model like nHibernate or Gentle.Net, or when its released using LINQ." ORM stands for Object-Relational Model, and is essentially an object-oriented way of dealing with table-based relational data. Here's a quote from the [nHibernate home page](#):

NHibernate is a port of Hibernate Core for Java to the .NET Framework. It handles persisting plain .NET objects to and from an underlying relational database. Given an XML description of your entities and relationships, NHibernate automatically generates SQL for loading and storing the objects. Optionally, you can describe your mapping metadata with attributes in your source code.

Now that sounds like just what the doctor ordered - a generalized way to work with SQL data in an object-oriented form. And with a layer like that in place you're definitely well on your way to a clean and maintainable multi-tier design. But I don't expect to see this in the first release of Clarion.NET.

I also suggest you do a little reading on [LINQ](#), which is Microsoft's approach to the same task. I've played with LINQ, and it's pretty cool stuff. Imagine being able to execute SQL-like queries not just on SQL and XML data, but on the objects in your running application....

Does multi-tier really matter?

The real question, for many developers, is whether this multi-tier architecture will matter to their clients. I think the answer is "it depends". If you're primarily developing client/server applications, then multi-tier isn't as important. After all, your existing Clarion apps have been doing the job nicely, right? In this case your reasons for going to Clarion.NET will have more to do with the vast library of .NET controls and libraries (in other words, new functionality) than with architecture. And presumably if you're sticking with the standard Clarion.NET templates you'll be doing some measure of multi-tier whether you know it or not.

If, however, you have a requirement for multi-tier development, or if you find yourself in the situation of needing to share a code base between desktop and web applications, there are clearly some compelling reasons to consider a multi-tier architecture. The more code you can share between applications, the lower your workload and the easier the maintenance.

Having mentioned web applications, I want to reinforce that in this article I am talking primarily about desktop and client/server apps. While many of the principles discussed here apply to web applications, there are some further design issues you should consider when developing for the web. I'll have more on that in a future article.

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, October 13, 2007 by John Dunn

Great article, Dave! I am very much looking forward to being able to develop multi-tier layer applications in Clarion.NET (or at least be more capable of doing so).

John

Posted on Saturday, October 13, 2007 by Stephen Ryan

Clarion right now supports both abstract programming and persistance.

We have for to long and with good reason not coded up abstract interfaced based systems.

In Clarion 6.3 you can start using multi layered Interfaced based systems that removed all aspects of the Clarion specific language syntax specifications.

Also since clarion 6 implements extensive XML supports you can implement your own XML object persistance.

Becuase we have an APP GEN we dont use clarion new language enhancements becuase we havent needed to.

In fact today in clarion you can program pretty much anything you can do in csharp and java.

Interfaces for :

Data Item Class

-----> Data Object Class

-----> Data Variable Management Class

-----> XML Persistance Class

----->Object Base Class

-----> Object Event arguments Class

-----> Object Event Class

-----> Control Interface Class

-----> Binding Source Class

-----> Queue Binding Class

-----> View Binding Class

ect, ect , ect

Its all there for you now.

But with the APP GEN and ABC its been easier just to get it out there.

An APP GEN that generate clarion dot net may well be all that needed for many projects so the need to develop highly complex systems may only represent a small percentage of clarion developments.

However it should be a lot easier to generate dot net apps in clarion dot net with an APP GEN than to write hand coded classes in csharp.

Dot net is a highly complex framework and clarion has always been about rad.

SO if you want to do OOP in clarion just like you do in Java or csharp you can.

But if you want to knock out an app using the APP GEN and the dictionary you can also go RAD.

Clarion caters for all tastes and styles, something that few other products do.

Clarion DOT NET will give developers the best of both worlds.

And im sure SV know this.

Posted on Monday, October 15, 2007 by Dave Harms

John,

Thanks, me too!

Steve,

>> In fact today in clarion you can program pretty much anything you can do in csharp and java.

I'd say that the lack of true introspection/reflection makes many tasks a lot more difficult, but yet, it's certainly possible to do a great deal more in Clarion than most developers realize.

Dave

Posted on Tuesday, October 16, 2007 by Stephen Ryan

Yes Clarion does not have those features.

However Jim kane as shown us in many articles that for a large number of advanced job a bit of assembly does the job.

The problem is so many of us dont kow our assembly and so Clarion DOT NET provides those tools at a higher level.

But thats why you should always have a engineer on hand for the low level stuff!

Posted on Tuesday, October 16, 2007 by Stephen Ryan

Also i have found that Evaluate is much under rated.

When combined with DLIB you can get a lot back from clarion.

DLIB supports native clarion declarations.

Evaluate and it new bind allows in conjunction with DLIB lets you at a hugh number of internal data structures.

If you absrtact your system and provide a procedure for access to all objects you can create some more flexition that is offered.

[Add a comment](#)

Clarion Magazine

PostgreSQL Revisited: Essential Tools

by Dave Harms

Published 2007-10-11

In the [first installment](#) in this series on [PostgreSQL](#) I showed how to install the PostgreSQL server on Windows (Vista, in my case). This time around I'll take a look at some of the administration tools included with PostgreSQL.

First, by way of review, let's look at what you've accomplished by installing PostgreSQL. These are the main components you'll have:

- The PostgreSQL server, installed as a service
- The psql text mode administration/query tool
- The pgAdmin III graphical administration/query tool
- Some SQL data - two tablespaces and at least one database
- The ODBC driver
- A whole bunch of command-line utilities; see the PostgreSQL bin directory

I'll summarize each of these, then look at how they work together to give you a complete database package.

The PostgreSQL server service

Whether you're installing PostgreSQL on a dedicated server, or on a workstation where you want to use it for local testing, the server itself will run as a service. So a couple of definitions are in order.

The PostgreSQL server is the program that mediates your program's access to the data. When you're using a flat file system like TPS files your program access the data directly - it reads the physical file and it writes the physical file. If there are multiple programs accessing a physical file at the same time then those programs have to do so in a manner that ensures data integrity is maintained.

SQL tables are, however, not accessed directly by the application. Instead the application requests data from a server, and the server reads and writes the physical file(s). You may have many applications accessing the data, but they all do so via requests to the server, which "serves up" the data and also updates data as requested.

The PostgreSQL server application is installed as a *service* on Windows, primarily so that it can be set to always start up along with Windows. Services don't need to interact with a user to run, and in fact they don't even need anyone to log on to the machine. That way you can have a dedicated PostgreSQL database server that's available simply because it's powered on.

So there you have it - a *server* running as a *service*. Now, how do you go about using the server?

Using PostgreSQL client software

There several ways you can interact with and administer a PostgreSQL database, including commercial and free interactive GUI tools (more about those in a moment). The most basic way, however, is to run the psql command line client. With psql you type a statement, and psql returns the result.

You'll see a shortcut to psql in your PostgreSQL start menu folder; Figure 1 shows the psql client at startup.

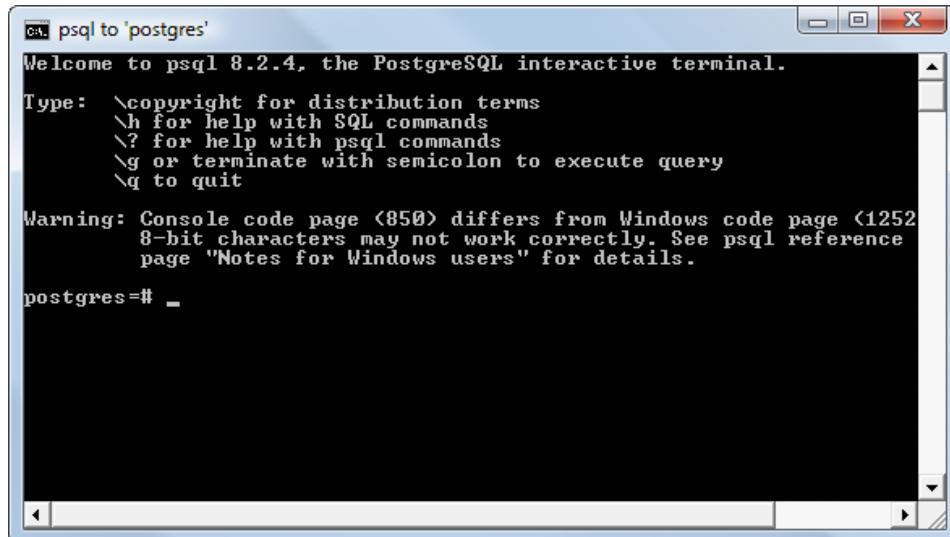
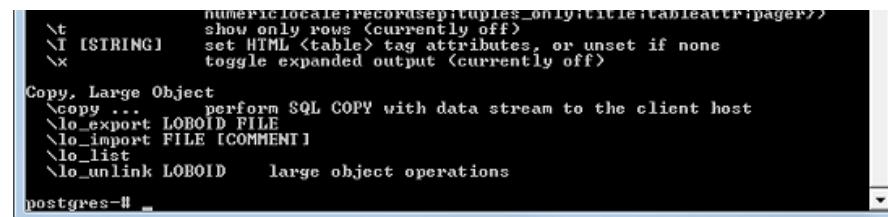


Figure 1. The psql client

When psql starts it displays a very basic list of commands, including a way to quit and a way to execute a multi-line query. But very importantly it's telling you that it can execute two different kinds of commands: administrative, and SQL. If you execute the \? command you'll see the screen shown in Figure 2.

The screenshot shows a Windows-style window titled "psql to 'postgres'". The title bar has standard minimize, maximize, and close buttons. The main area contains a detailed list of psql commands categorized by type:

- General**
 - \c [DBNAME] - USER!- HOST!- PORT!- connect to new database (currently "postgres")
 - \cd [DIR] change the current working directory
 - \copyright show PostgreSQL usage and distribution terms
 - \encoding [ENCODING] show or set client encoding
 - \h [NAME] help on syntax of SQL commands, * for all commands
 - \q quit psql
 - \set [NAME] [VALUE] set internal variable, or list all if no parameters
 - \timing toggle timing of commands (currently off)
 - \unset NAME unset <delete> internal variable
 - \! [COMMAND] execute command in shell or start interactive shell
- Query Buffer**
 - \e [FILE] edit the query buffer (or file) with external editor
 - \g [FILE] send query buffer to server (and results to file or |pipe)
 - \p show the contents of the query buffer
 - \r reset <clear> the query buffer
 - \w FILE write query buffer to file
- Input/Output**
 - \echo [STRING] write string to standard output
 - \i FILE execute commands from file
 - \o [FILE] send all query results to file or |pipe
 - \qecho [STRING] write string to query output stream (see \o)
- Informational**
 - \d [NAME] describe table, index, sequence, or view
 - \ddt[ilisv]s[PATTERN] [PATTERN] (add "+" for more detail)
 - list tables/indexes/sequences/views/system tables
 - list aggregate functions
 - list tablespaces (add "+" for more detail)
 - list conversions
 - list casts
 - show comment for object
 - list domains
 - list functions (add "+" for more detail)
 - list groups
 - list schemas (add "+" for more detail)
 - list operators
 - list large objects, same as \lo_list
 - list table, view, and sequence access privileges
 - list data types (add "+" for more detail)
 - list users
 - list all databases (add "+" for more detail)
 - list table, view, and sequence access privileges (same as \dp)
- Formatting**
 - \a toggle between unaligned and aligned output mode
 - \c [STRING] set table title, or unset if none
 - \f [STRING] show or set field separator for unaligned query output
 - \h toggle HTML output mode (currently off)
 - \pset NAME [VALUE] set table output option
 - (NAME := {format|border|expanded|fieldsep|footer|null|numericlocale|recordsep|tuples_only|title|tableattr|pager})
 - \rt show only rows (currently off)
 - \ti [STRING] set HTML <table> tag attributes, or unset if none
 - \x toggle expanded output (currently off)
- Copy, Large Object**
 - \copy ... perform SQL COPY with data stream to the client host
 - \lo_export LOBOID FILE
 - \lo_import FILE [COMMENT]
 - \lo_list
 - \lo_unlink LOBOID large object operations



```

\h [STRING]      show only rows (currently off)
\i [table]       set HTML <table> tag attributes, or unset if none
\o [file]        toggle expanded output (currently off)

Copy, Large Object
\copy ...        perform SQL COPY with data stream to the client host
\lo_export LOBOID FILE
\lo_import FILE [COMMENT]
\lo_list
\lo_unlink LOBOID    large object operations

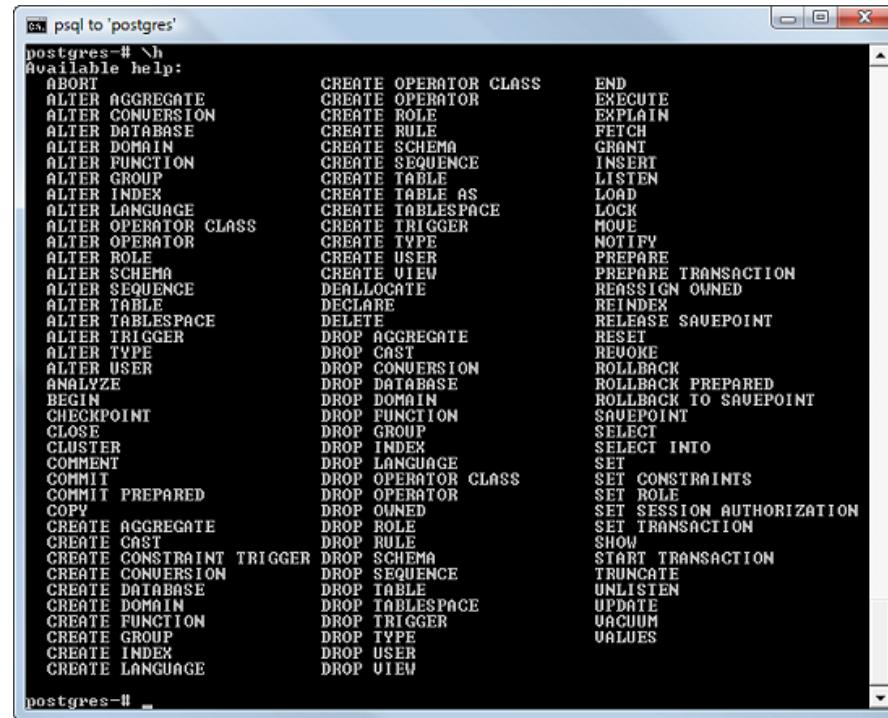
postgres=#

```

Figure 2. Administrative commands ([view full size image](#))

Among other things, you use the administrative commands to connect to databases, list information about those databases, and change display formatting options.

When it comes to actually working with SQL data, however, you'll use SQL commands. For a list of the SQL commands PostgreSQL understands, type `\h`, as in Figure 3.



```

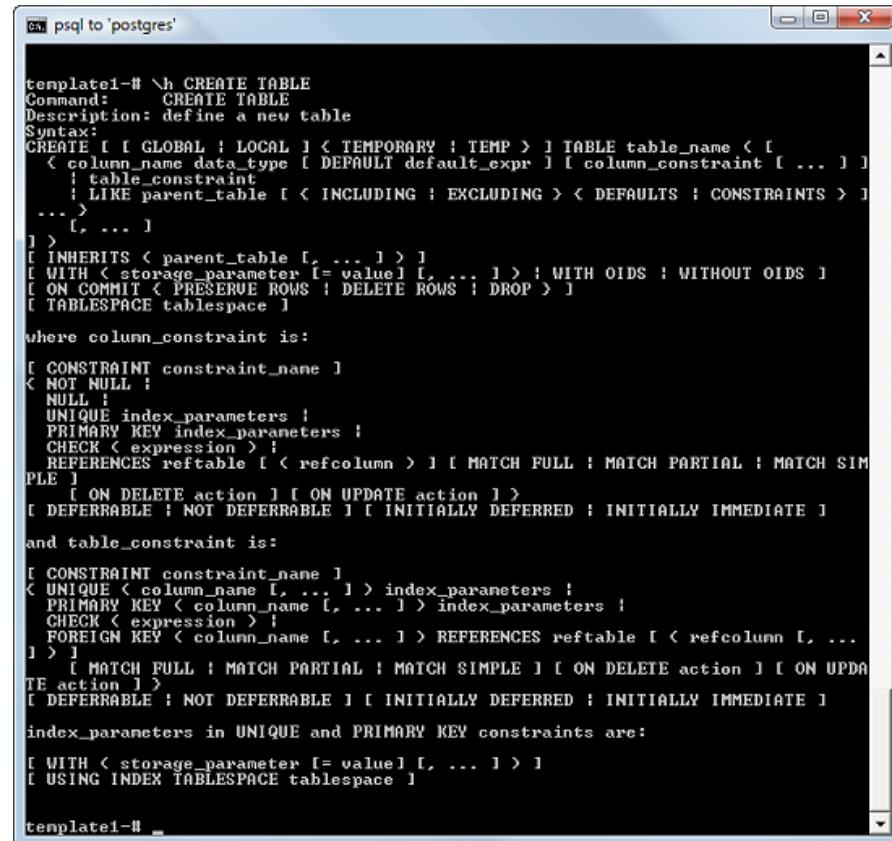
psql to postgres
postgres=# \h
Available help:
  ABORT           CREATE OPERATOR CLASS   END
  ALTER AGGREGATE CREATE OPERATOR          EXECUTE
  ALTER CONVERSION CREATE ROLE             EXPLAIN
  ALTER DATABASE   CREATE RULE            FETCH
  ALTER DOMAIN    CREATE SCHEMA          GRANT
  ALTER FUNCTION   CREATE SEQUENCE        INSERT
  ALTER GROUP     CREATE TABLE           LISTEN
  ALTER INDEX     CREATE TABLE AS        LOAD
  ALTER LANGUAGE   CREATE TABLESPACE      LOCK
  ALTER OPERATOR CLASS CREATE TRIGGER      MOVE
  ALTER OPERATOR   CREATE TYPE           NOTIFY
  ALTER ROLE       CREATE USER           PREPARE
  ALTER SCHEMA    CREATE VIEW           PREPARE TRANSACTION
  ALTER SEQUENCE   DEALLOCATE          REASSIGN OWNED
  ALTER TABLE     DECLARE              REINDEX
  ALTER TABLESPACE DELETE               RELEASE SAVEPOINT
  ALTER TRIGGER   DROP AGGREGATE        RESET
  ALTER TYPE      DROP CAST             REVOKE
  ALTER USER      DROP CONVERSION      ROLLBACK
  ANALYZE         DROP DATABASE        ROLLBACK PREPARED
  BEGIN           DROP DOMAIN          ROLLBACK TO SAVEPOINT
  CHECKPOINT     DROP FUNCTION         SAVEPOINT
  CLOSE           DROP GROUP           SELECT
  CLUSTER         DROP INDEX           SELECT INTO
  COMMENT         DROP LANGUAGE          SET
  COMMIT          DROP OPERATOR CLASS   SET CONSTRAINTS
  COMMIT PREPARED DROP OPERATOR          SET ROLE
  COPY            DROP OWNED           SET SESSION AUTHORIZATION
  CREATE AGGREGATE DROP ROLE            SET TRANSACTION
  CREATE CAST      DROP RULE             SHOW
  CREATE CONSTRAINT TRIGGER DROP SCHEMA
  CREATE CONVERSION DROP SEQUENCE        START TRANSACTION
  CREATE DATABASE  DROP TABLE           TRUNCATE
  CREATE DOMAIN   DROP TABLESPACE      UNLISTEN
  CREATE FUNCTION  DROP TRIGGER          UPDATE
  CREATE GROUP    DROP TYPE            VACUUM
  CREATE INDEX    DROP USER             VALUES
  CREATE LANGUAGE  DROP VIEW          

postgres=#

```

Figure 3. SQL commands ([view full size image](#))

For detailed help, simply type `\h` followed by the SQL statement. Figure 4 shows the help for `CREATE TABLE`.

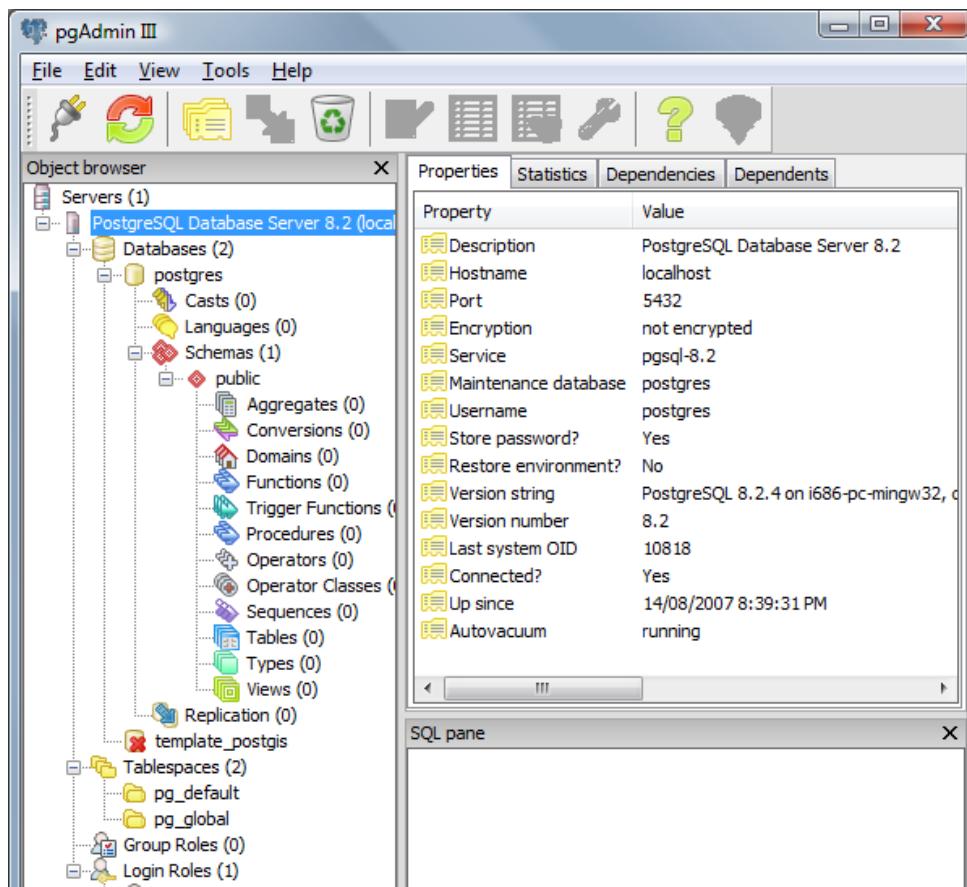


```
psql to 'postgres'
template1=# \h CREATE TABLE
Command: CREATE TABLE
Description: define a new table
Syntax:
CREATE [ [ GLOBAL | LOCAL ] < TEMPORARY | TEMP > ] TABLE table_name < [
    { column_name data_type [ DEFAULT default_expr ] [ column_constraint [ ... ] ]
    | table_constraint
    | LIKE parent_table [ < INCLUDING | EXCLUDING > < DEFAULTS | CONSTRAINTS > ]
    ...
    | ...
] >
[ INHERITS < parent_table [ , ... ] > ]
[ WITH < storage_parameter [= value] [, ... ] > | WITH OIDS | WITHOUT OIDS ]
[ ON COMMIT < PRESERVE ROWS | DELETE ROWS | DROP > ]
[ TABLESPACE tablespace ]
where column_constraint is:
[ CONSTRAINT constraint_name ]
< NOT NULL ;
NULL ;
UNIQUE index_parameters ;
PRIMARY KEY index_parameters ;
CHECK < expression > ;
REFERENCES reftable [ < refcolumn > ] [ MATCH FULL | MATCH PARTIAL | MATCH SIMPLE ]
[ ON DELETE action ] [ ON UPDATE action ]
[ DEFERRABLE | NOT DEFERRABLE | INITIALLY DEFERRED | INITIALLY IMMEDIATE ]
and table_constraint is:
[ CONSTRAINT constraint_name ]
< UNIQUE < column_name [, ... ] > index_parameters ;
PRIMARY KEY < column_name [, ... ] > index_parameters ;
CHECK < expression > ;
FOREIGN KEY < column_name [, ... ] > REFERENCES reftable [ < refcolumn [, ...
] > ]
[ MATCH FULL | MATCH PARTIAL | MATCH SIMPLE ] [ ON DELETE action ] [ ON UPDATE action ]
[ DEFERRABLE | NOT DEFERRABLE | INITIALLY DEFERRED | INITIALLY IMMEDIATE ]
index_parameters in UNIQUE and PRIMARY KEY constraints are:
[ WITH < storage_parameter [= value] [, ... ] > ]
[ USING INDEX TABLESPACE tablespace ]

template1# _
```

Figure 4. Help for CREATE TABLE (view full size image)

If you're not comfortable using the command line, there are a number of free and commercial GUI interfaces which ease administrative tasks. PostgreSQL comes with the pgAdmin III client, as seen in Figure 5.



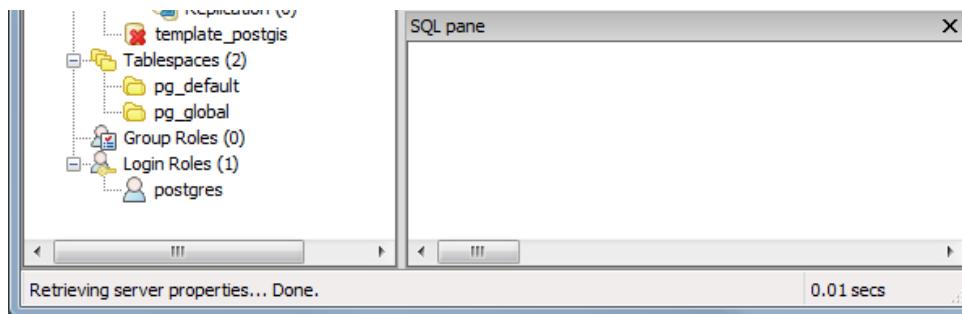


Figure 5. The pgAdmin III administrative tool

pgAdmin III does a decent job of letting you create and administer tablespaces, databases, users and roles, but it isn't the most sophisticated tool for the job. I've also used [EMS SQL Manager for PostgreSQL](#); if you're considering a commercial product this one is worth a look.

As I continue on in this series I'll provide both psql and pgAdmin III examples of the various administrative tasks, where appropriate.

With your choice of admin tool you're ready to start creating users, roles, tablespaces and databases. Users and roles relate to security issues and I'll touch on those next time; first, let's talk about the data.

Clusters, tablespaces, and databases

In part one I noted that the install program lets you initialize a database cluster, which can contain one or more databases. I also implied that a tablespace and a cluster are the same thing.

Here's the official definition of a PostgreSQL cluster:

Tables are grouped into databases, and a collection of databases managed by a single PostgreSQL server instance constitutes a database cluster.

So where does the concept of a tablespace fit in? Again, from the Help:

Tablespaces in PostgreSQL allow database administrators to define locations in the file system where the files representing database objects can be stored. Once created, a tablespace can be referred to by name when creating database objects.

By using tablespaces, an administrator can control the disk layout of a PostgreSQL installation. This is useful in at least two ways. First, if the partition or volume on which the cluster was initialized runs out of space and cannot be extended, a tablespace can be created on a different partition and used until the system can be reconfigured.

Second, tablespaces allow an administrator to use knowledge of the usage pattern of database objects to optimize performance. For example, an index which is very heavily used can be placed on a very fast, highly available disk, such as an expensive solid state device. At the same time a table storing archived data which is rarely used or not performance critical could be stored on a less expensive, slower disk system.

Really when the install program asks you to create a cluster, what it is doing is creating a tablespace with at least one database containing some system tables. So should you rush out and create a bunch of new tablespaces? If you don't have a specific reason, such as those indicated above, probably not. Also from the help:

There is usually not much point in making more than one tablespace per logical file system, since you cannot control the location of individual files within a logical file system.

The installation program creates two tablespaces for you, in apparently flagrant violation of the help's advice. But that's because one of those tablespaces is for system information - it's called pg_global and you should leave it alone. The other tablespace is pg_default and you can put your own databases in there.

Default databases

When you create a new database in PostgreSQL (either via a GUI tool or by means of CREATE DATABASE) you do so by making a copy of an existing database. There are two *template* databases included with PostgreSQL, template0 and template1. There's nothing particularly special about these databases; they're simply empty databases. As well, they're identical, at least as of the point of installation.

The difference between template0 and template1 is that you may not modify template0, but you can make changes to template1. And template1 is the default template used when you create a new database. You can specify any existing database as the template for a new database, however. If you've made changes to template1 and you want to create a new, completely empty database, you can specify template0 as the new database's template.

You may be thinking that a database contains simply tables, but as with many SQL servers, PostgreSQL's databases can also contain things like triggers, functions (stored procedures), and even language extensions such as special data types and operators. If, for instance, you have some standard triggers or stored procedures you always use with your databases, you may want to add them to template1 so they're automatically copied to your new databases.

The ODBC driver

There are a number of ways applications can connect to PostgreSQL, but for Clarion Win32 applications the most likely choice is the ODBC driver, which is included with PostgreSQL. There is also at least one [commercial ODBC driver](#) for PostgreSQL.

As with other ODBC drivers, your application doesn't talk to the PostgreSQL driver directly. Instead you set up your tables to use the SoftVelocity ODBC file driver, and specify the data source that corresponds to that table. You configure a data source in the ODBC administrator, and it's that data source that uses the PostgreSQL driver. So your application talks to the SoftVelocity ODBC file driver, the SV driver talks to the PostgreSQL ODBC driver, and the PostgreSQL ODBC driver talks to the database. I'll walk through some examples in future articles.

Application interfaces

If ODBC isn't good enough for you, you can always take a stab at the C library interface, libpq. You'll have to create a LIB with LibMaker and prototype the desired functions for Clarion (if anyone's done the prototyping already and is willing to share their work, please post a reader comment below). The library can be used synchronously (your program waits until the command completes) or asynchronously (your program submits a command, then continues to respond to the user while waiting for a result back from the server - which is generally preferable).

The command line utilities

There are a number of command-line utilities included with PostgreSQL. For the most part these are wrappers for commonly-used SQL statements, but there are some additional administrative programs you should know about.

The utilities are in the bin directory and include:

- clusterdb.exe - For physically sorting tables based on index values for performance reasons. You can also do this via the SQL CLUSTER statement. This is a one-time sort; further changes to the data will not be in index order and you may need to redo the clustering periodically. This is a specialized performance technique and will probably not be that useful in most business applications.
- createdb.exe - Create a new database. This is a wrapper for the SQL CREATE DATABASE command.
- createlang.exe - A wrapper for the CREATE LANGUAGE command. This registers a procedural language with the PostgreSQL server. PostgreSQL has hooks that let you create your own procedural language, if you wish. For more information see the PostgreSQL documentation, [Chapter 47. Writing A Procedural Language Handler](#).
- createuser.exe - Create a new user. This is a wrapper for the CREATE ROLE command.

- dropdb.exe - Drop a database. This is a wrapper for the DROP DATABASE command.
- droplang.exe - Drop a language. This is a wrapper for the DROP LANGUAGE command.
- dropuser.exe - Remove a user. This is a wrapper for the DROP ROLE command.
- initdb.exe - Create a new database cluster, which is basically a collection of databases in one or more tablespaces. In other words it's the set of data managed by a single instance of the database server. Normally this initialization is done during the installation process but you can initialize a new cluster manually via this utility.
- oid2name.exe - Show various internal information about database objects. You can use the command oid2name -? to get a help screen.
- pgAdmin3.exe - The executable file for pgAdmin III.
- pgAgent.exe - A [utility](#) for scheduling database jobs, such as maintenance tasks.
- pgbench.exe - A tool for benchmarking/stress testing PostgreSQL.
- pg_config.exe - Prints configuration information about PostgreSQL. Again, use the -? parameter for a help screen.
- pg_controldata.exe - Prints control information for the data directory (i.e. the database cluster) including locale and checkpoint processing and write-ahead logging.
- pg_ctl.exe - Lets you start, stop, and restart a PostgreSQL server, or display the status of a running server. Most likely PostgreSQL is already installed as a service on your machine and is started automatically, so you won't need to use this utility.
- pg_dump.exe - Dump a database either to a text file (plain text SQL statements) or to an archive file readable by pg_restore. This utility is non-blocking; reads/writes will continue while the data is being dumped. There are a variety of options including dumping schemas only (no data), data only (not schemas), dump only specified schemas or exclude certain schemas, include or exclude tables, etc. The options are extensive and make this utility suitable for a variety of backup techniques as well as creating SQL scripts to import data into non-PostgreSQL databases.
- pg_dumpall.exe - Like pg_dump, but works on the entire cluster.
- pg_resetxlog.exe - A utility of last resort for fixing a corrupted write-ahead log.
- pg_restore.exe - Restore an archive (non-text) file created with pg_dump.
- postgres.exe - the PostgreSQL server itself. Use pg_ctl instead.
- psql.exe - The psql command line client (you already have a shortcut to this on your Start menu).
- vacuumdb.exe - A wrapper for the SQL VACUUM statement. VACUUM analyzes or cleans up a database. In PostgreSQL when a record is deleted that space is not automatically reused; similarly, updated records are written to a new location, and the old record continues to use up space. This makes sense from a transactional security point of view, but in heavily modified tables you can end up with a lot of dead space. There are several different ways to clean up the database. As of PostgreSQL 8.1 there is an autovacuum daemon; look in postgres.conf for the # AUTOVACUUM PARAMETERS section. If you want or need to run VACUUM manually, the VACUUM FULL command is the most aggressive form and will return unused space to the operating system, but requires table locks and can have a significant impact on performance; a plain or "lazy" VACUUM does not return unused space but simply marks deleted rows as available for use. For most databases without the autovacuum daemon, VACUUM should be run once per day (you may want to schedule a job for this - see pgAgent above). Tables with intensive use may benefit from more frequent VACUUMING. VACUUM can also run an ANALYZE which collects statistical data used to optimize data retrieval. ANALYZE doesn't have to be done as often as VACUUM unless you have large tables where the distribution of data is atypical and changes regularly. The autovacuum daemon also runs ANALYZE, so if you're on 8.1 or later you may not need to run ANALYZE at all.
- vacuumlo.exe - Removes any orphaned large objects from the database.

Summary

A basic Windows PostgreSQL installation includes the server (installed as a service), a database cluster containing

two tablespaces and some standard databases, the psql and pgAdmin III client programs, the ODBC driver, and some useful utility programs.

With your PostgreSQL server up and running, the next step is to make some choices about who can access which database and for what purpose. I'll cover user authentication 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

The ClarionMag Blog

Get automatic notification of new items! RSS feeds are available for:

[XML All blog entries](#)

[XML All new items, including blogs](#)

Blog Categories

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

MVC web apps, Microsoft, and Clarion.NET

[Direct link](#)

Posted Friday, October 12, 2007 by Dave Harms

In yesterday's [article](#) on .NET application architecture I alluded to web application design issues and indicated I'd be covering that topic in a future article. I will say now that I've long been a fan of the Model-View-Controller (MVC) approach. In fact, the Clarion Magazine web server, which I wrote in Java, is built on an MVC design, and for some time now I've been interested in porting it over to .NET. ASP.NET isn't really an MVC framework (although if you beat it with a hammer you can get it roughly into shape) so I've been considering the Castle Project's [Monorail](#), a Ruby-on-Rails-inspired framework for .NET. Now Microsoft has jumped on the MVC web app bandwagon with a project it expects to release around the end of the year, in beta. Microsoft's Scott Guthrie [gave a presentation](#) earlier this month at the Alt.net conference in Austin, TX. Ruby fans take note - this project is getting some [good buzz](#). And apparently the Monorail folks aren't too worried as MS is building the MVC framework, while [Monorail has that and more](#). The MS MVC code could be integrated into Monorail.

You can watch Scott's presentation [online](#), but be warned; this is hand held camera work and it's none too steady. Toward the end of the presentation I found myself just about ready to hurl, which was a little ironic because for a change I think MS is on to something good with this project.
