Clarion Magazine

This edition includes all articles, news items and blog posts from December 1 2010 to December 31 2010.

Clarion News

Read 19 Clarion news items.

Articles

MagGems #1, #2 and #3

December 2 2010

If you haven't been to a ClarionLive webinar recently you're missing out on MagGems, featured articles from Clarion Magazine's archives. Dave Harms summarizes the first three MagGems.

ClarionMag Tip of the Week #2: Cleaning Builds

December 8 2010

Any Clarion compilation creates a bunch of code that you don't really need, including object files, resource files, executables and DLLs. Here's a quick and easy way to delete those files.

ClarionMag at ClarionLive: A Webinar That Could Completely Change How You Code

December 9 2010

As developers, we're paid to write code. And not just any code, but the good stuff, quality code that works reliably. How can you know your code is reliable? By testing it, of course. On Friday, Dec 10, in a ClarionLive webinar, Dave Harms will introduce an updated version of the ClarionTest unit testing framework and show how to use ClarionTest to make your code not just reliable but also more maintainable and reusable.

ClarionMag Tip Of the Week #3: Use Directory Symlinks To Navigate To C7

December 15 2010

Clarion 7's default installation directory can be a hassle when you need to navigate to, say, the libsrc directory. One alternative is to install to a shorter directory name like C:\Clarion7. A more flexible option is to create a symbolic directory link.

Merry Christmas! ClarionMag's Holiday Schedule And Upcoming Articles

December 23 2010

The Clarion Magazine office will be closed Friday, December 24th. Here's a look at what's still to come this year.

ClarionMag Tip of the Week #4: Double Your App's Available Memory On Win64

December 24 2010

Are you bumping up against your 32 bit application's 2GB memory limitation? If you have Clarion 7.1 or later, this one line of code can give you access to up to 4GB of RAM on 64 bit Windows.

The Best Thing About Clarion 7

December 27 2010

Steve Parker was just plain gobsmacked when he realized he had found something in C7 that improved his code and did so with virtually no effort on his part.

The Birth Of A Compendium

December 28 2010

A compendium can be defined as 'a concise, yet comprehensive compilation of a body of knowledge'. Robert Barton shows how he created a compendium for his client in just two weeks, using a handful of key third party products.

ClarionMag Tip Of The Week #5: Zoom In On Small Prompts

December 31 2010

You've seen them in the AppGen: teeny tiny prompts too small to display all the text they hold. Happily, there's an easy way to make those prompts big enough to display all the text at once.

Start Testing Your Clarion Win32 Code With ClarionTest

December 31 2010

Unit testing and test-driven development have the potential to transform how you work and greatly increase the quality of your code. Dave Harms introduces ClarionTest 0.3, with some important productivity improvements.

Clarion News

EZChangeLog Reporter V1.8

EZChangeLog Reporter adds additional report and data query capabilities, including output to Microsoft Excel spreadsheets for LansRad's EZChangeLog Professional data files. Version 1.8.2.788 fixes a Windows Alt-key freeze problem.

Posted December 1 2010 (permanent link)

Icetips Utilities 1.1.2392

Icetips Utilities 1.1.2392 is now available. This build is mostly updates to the documentation, which has now reached 400 pages in the PDF file.

Posted December 2 2010 (permanent link)

List & Label On Azure

Combit has optimized the development component List & Label for the cloud computing platform Windows Azure. A white paper is available for download.

Posted December 6 2010 (permanent link)

C7.3 Pre-release, DLL Versioning Changes

A pre-release of C7.3 has gone out, with a notable change in how DLL naming and versioning is done. The internal version checking code now uses an internal RTL version as well as the DLL version to reduce the need for complete rebuilds.

Posted December 6 2010 (permanent link)

SetupBuilder 7.3 Release Announcement

Lindersoft has released SetupBuilder Version 7.3, the latest edition of its award-winning Installation Authoring and Configuration Management system for Microsoft Windows based applications. This release is available, free of charge, to all SetupBuilder customers who have an active SetupBuilder maintenance and support subscription plan. Among other features this release adds support for the Clarion 7.3 environment.

Posted December 6 2010 (permanent link)

DMC Gold 2.3.2.1231

DMC Gold version 2.3.2.1231 is now available. Changes include: released 8th. December

2010 ; ; Port your Application to SQL added a Wizard like Step by Step to allow you an easier navigation for this feature ; Port your Application to SQL added the possibility for you to DEPLOY your new SQL Tables to your end users machines (with the Runtime Engine) ; Port your Application to SQL added support for more c6 to c7 DCT errors (GLOBALS without a DRIVER - OPTION control with several predefined values etc ...) ; Port to SQL - TXA parsing added many Legacy "cases" and during table Name matching (Create profiles) a NEW display order to make selections easier. ; Port to SQL - Save your settings to HD and Restore settings from HD ; Changed code used when Transferring Data to FireBird with BLOBS and on Alias Tables in Port to SQL ; Added in Settings the possibility to access all SQL Normalization options and to define the CASE to use for DCTX and TXD generation ; Added in Settings the possibility to display or not at the end of a data transfer the records (save time if not needed) ; Corrected several bugs in Clone to SQL (Firebird) when a BLOB NON Binary column existed in Table or when a DynIndex is the last key.

Posted December 17 2010 (permanent link)

EasyCOM2INC 2.12

EasyCOM2INC 2.12 is now available. Changes include: For the new project flag 'Link oleautcg.lib' will be set automatically to TRUE for Clarion < 7.1, and to FALSE for Clarion >= 7.1; Fix for the dispinterface - with the huge number of methods not all methods were generated in the interface section; Fix for generator where parameters with the [out] attribute could not return the new value to the caller; Generator XML comments will be generated for the methods which have [helpstring] attribute (to use with Code Completion in C7). Price: $189.

Posted December 17 2010 (permanent link)

CapeSoft C7.3 Builds

Clarion 7.3 builds of CapeSoft accessories are now available for download.

Posted December 17 2010 (permanent link)

CapeSoft Retires Clarion 5, NetTalk 3

Capesoft is officially saying good bye to two ol' faithful friends: Clarion 5 and NetTalk 3. As of 31 December 2010, CapeSoft will no longer provide builds for Clarion 5 (for the DLL based products) and no more option for C5 in the installs. C7.3 will be the last supported build for NetTalk 3. From that point on CapeSoft will only be building NetTalk 4 and 5. NetTalk 4 is now 5 years old.

Posted December 17 2010 (permanent link)

Icetips Taskpanel for C7.3

A new build of Icetips Taskpanel available for download. This build is compatible with Clarion 7.3 and includes some minor fixes. The Icetips Taskpanel is part of the Icetips Gold

Subscription.

Posted December 17 2010 (permanent link)

ClarioNET Android client demo now available.

Following the suggestion put forward during the ClarioNET webinar, the ClarioNET Android client is now also available in demo mode (fully working but cuts out every 5 minutes). Anyone wishing to have a pre-release copy to test on their machines may obtain a copy by sending an email to ivanmintoff@clarionet.com with the header "ClarioNET Android".

Posted December 17 2010 (permanent link)

PD Class Generator Version 7.2.0004

An updated version of PD Class Generator is now available. It includes a bug fix and several enhancements, among them a new useful example, a Window Manager Class which maintains a queue of open windows. The Queue includes the procedure name, a reference to the window, its thread, and whether it is modal or not. It includes methods for posting events and notifications to windows by procedure name, keeping windows in the Application client area when it is resized, and more.

Posted December 20 2010 (permanent link)

Icetips C7 Builds

All Icetips products are now Clarion 7.3 compatible.

Posted December 31 2010 (permanent link)

EasyNaviBar 1.04

EasyNaviBar 1.04 is now available. Changes include: Options > Add or remove buttons wasn't working, fixed; New Save Settings / Load Settings in template and new SaveSettings / LoadSettings methods; Possibility of changing the standard text of Options menu, new ChangeOptionsMenuText method; Possibility of setting Options menu visibility, new SetOptionsMenuVisible method; Possibility to use images linked into exe file (~IconFile.ICO). This is a free upgrade for all customers who have a current (valid) subscription plan.

Posted December 31 2010 (permanent link)

Data Conversion Template 1.88

Data Conversion Template version 1.88 is now available. Changes include: Improved C7 support - prevented exception in some cases, also fixed the problem with groundless growing of svi-files for the tables with GROUPs; Fixed the issue which can occur during the conversion in different database contexts for one session.

Posted December 31 2010 (permanent link)

iQ-Notes Adds Tabs To Post It Notes

A new version of iQ-NOTES is available, with one big new feature: Post it notes can now have tabs. Other changes include: Ability to Import and Export Address Book so you copy easily on multiple computers; Added Ctrl+E / PopUp option to quickly Insert an Expense Tracking Item; Show Total Notes on About screen; Fixes to General Error Window; Updates to missing entries in Language tables; More improvements to resize logic on notes screen; Made StayOnTop and then Non StayOnTop better and no longer is a spawned application running; Sending Notes would not work if the Port was left as zero; Password fields for Sending Notes is now masked. iQ-Notes is a free "Post-It note" program for your PC.

Posted December 31 2010 (permanent link)

Ingasoftplus Time Limited New Year Offer: 25% Relative Discount

Ingasoftplus announces the cutting of prices and a 25% discount on all products (full versions only) for the period from 01 Dec till 31 Dec 2010.

Posted December 31 2010 (permanent link)

Shawn Mason SQL Training Early Bird Special Ends Dec 31

The early bird special of only $899 ($799 for online streaming attendance) ends Friday, December 31st. Hurry and get one of the last seats available. The Clarion/SQL seminar in Atlanta for Jan 27th-29th is almost full. You'll get three days of intensive SQL training inside the Clarion environment as well as help in changing your app over to SQL. Learn great tips and tricks as well as how the SQL engines function internally so you can make the best coding decisions possible.

Posted December 31 2010 (permanent link)

New Year Sale from 1st Logo Design

1st Logo Design is running a New Year's special. Icon collections are $39 each; when you buy 1 you get 1 free.

Posted December 31 2010 (permanent link)

The ClarionMag Blog

MagGems #1, #2 and #3

By Dave Harms

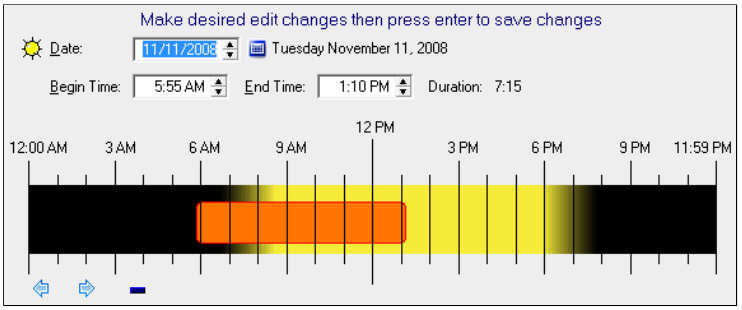
Posted December 2 2010

If you've ever taken part in a ClarionLive webinar you'll know that Arnold Young and John Hickey have a regular segment they call WebGems, in which they talk about interesting web pages and products, often but not always related in some way to the life of the Clarion developer.

Last month John and Arnold asked me to do a regular segment on the show called MagGems, each week featuring a Clarion Magazine article from the past. I've done three MagGems already, and I'll briefly go over them here.

MagGem#1: The time picker

Back in 2008 Paul Blais wrote an article titled Prompting For Time, in which he demonstrated a very clever user interface for selecting blocks of time using mouse click and drag. Paul's code is comprehensive, including location-specific calculations for sunrise and sunset in official, civil, nautical and astronomical time.



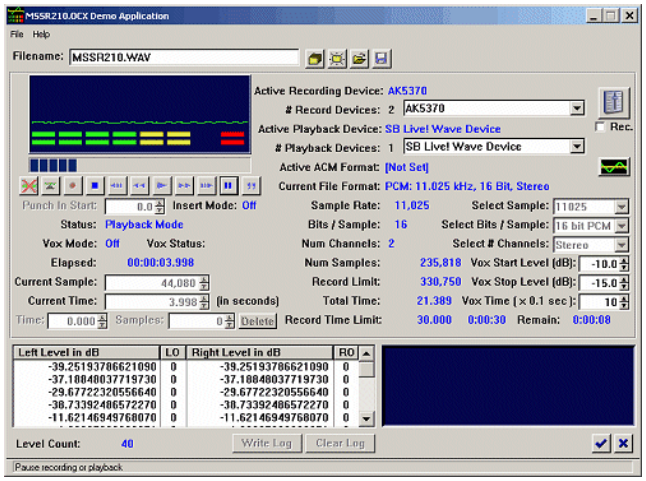
A C7 version of the application is available for download on the article page.

Watch the ClarionLive webinar containing this MagGem.

MagGem #2: Recording audio

The second MagGem goes all the way back to 2004, when Ben Brady wrote Recording Audio: An Introduction To OCXs, a three part series on how to record audio using the MSSR210 OCX. Ben's article is still an excellent introduction to OCXs, and John pointed

out that he'd implemented this code to let his customers add audio notes to an application.

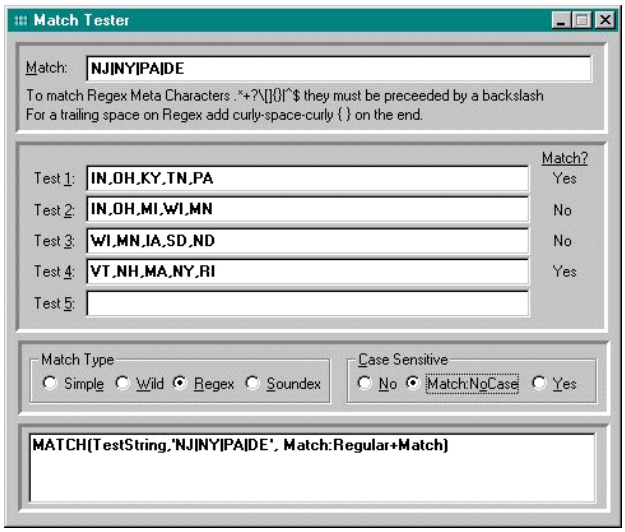


The C7 version of the source code uses the MSSR300 OCX, which is now freely available.

Watch the ClarionLive webinar containing this MagGem.

MagGem #3: Regular expressions

Regular expressions (regex) are a common feature of programming languages, but they are a later addition to the Clarion toolset. In Using MATCH In Filters and Regular Expressions Carl Barnes explains how to use the Match and StrPos functions to replace complex InString expressions. Once again there's a C7 version of the source.



I also demonstrated how I use regex in C7 to simplify comparing SQL schemas.

Watch the ClarionLive webinar containing this MagGem.

More to come

Tune in to ClarionLive on Fridays for more weekly MagGems; I'll also be posting MagGem summaries here in ClarionMag.

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).

Article comments

BACK TO TOP

ClarionMag Tip of the Week #2: Cleaning Builds

By Dave Harms

Posted December 8 2010

Any Clarion compilation creates a bunch of code that you don't really need, including object files, resource files, executables and DLLs. Okay, you need them to build and run your program, but you don't need them if you just want to preserve the *ability* to build and run your program. If you're making archival copies, checking files into a version control system or just sending an app over to another developer, there's no need to include all that extra stuff.

In C6 you have to delete all those files manually. But in C7 there's a new option on the top level Build menu: Clean.

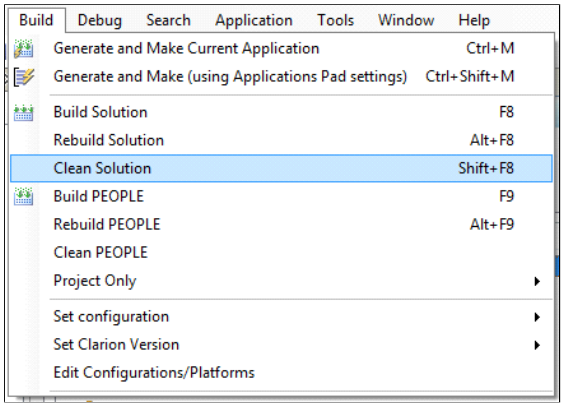


Figure 1. Cleaning a solution

Clean removes all the temporary compile/link files, as well as any DLLs copied to the directory as part of the build process and any DLL(s) or EXE(s) created by the build. Clean does *not*, however, delete generated CLW files, at least at this time. Those files are marked with a `<Generated>true</Generated>` element so perhaps there's a way of cleaning them that I haven't yet discovered. At the very least it should be possible to create a custom build task that would remove generated files *and* trigger a Clean.

Clean can free up a lot of disk space. I built a C7 version of the People app, and before cleaning it took up 4.6M in 60 files. After cleaning it was down to 716K in 20 files.

Note that you can clean either the entire solution or just the currently selected project. You can also right-click on the project entry in the Solution Explorer and clean a project from that context menu.

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).

Article comments

by Benjamin Dell on December 14 2010 (comment link)

Be carefull that you don't clean the solution if you intend to clean only one app.

I often want to clean a single app only and not the whole solution, and end up having to re-compile the complete 20+ app solution because the *.lib files are not available any more.

Kind Regards

Ben

---

by Dave Harms on December 14 2010 (comment link)

Good point Ben, thanks.

BACK TO TOP

ClarionMag at ClarionLive: A Webinar That Could Completely Change How You Code

Posted December 9 2010

As developers, we're paid to write code. And not just any code, but the good stuff, quality code that works reliably.

How can you know your code is reliable? By testing it, of course.

Unit testing and test-driven development are two incredibly important concepts that are reshaping the way programmers work. By following these practices you can produce not just more reliable code, but also more maintainable and reusable code.

You've probably heard about (or used) unit testing frameworks for newer platforms like .NET and Java. But what about Win32 coding? Can Clarion Win32 developers do unit testing and test-driven development too?

Absolutely we can. On Friday, December 10 2010 I will be presenting a ClarionLive webinar on unit testing, test-driven development, and isolating business logic into testable, reusable classes. I'll introduce a new, more intuitive version of the ClarionTest unit testing framework. I'll demonstrate an easy-to-use process for creating reusable Clarion classes. And I'll show how using a test-first approach actually makes it easier to design your classes.

I hope you'll join me for what I believe will be the beginning of an important change in how Clarion developers write code.

Seating is limited. At present the webinar is limited to 100 attendees, so get there early. And if you're unable to attend, either because the webinar is full or because of other obligations, the recording will be available for download later.

And look for follow-up articles in Clarion Magazine in the weeks ahead.

Article comments

BACK TO TOP

ClarionMag Tip Of the Week #3: Use Directory Symlinks To Navigate To C7

By Dave Harms

Posted December 15 2010

The default location for Clarion 7 installs is under C:\Program Files\SoftVelocity or (if you're running a 64 bit version of Windows) under C:\Program Files (x86)\SoftVelocity. And that makes navigating to the libsrc and templates folders somewhat painful. It's even worse if you do as I do and install each version of C7 separately. As Figure 1 indicates, if I want to go to the libsrc directory for C7.3 I need to nagivate all the way to C:\Program Files (x86) \SoftVelocity\Clarion7.3.7852\LibSrc\win.

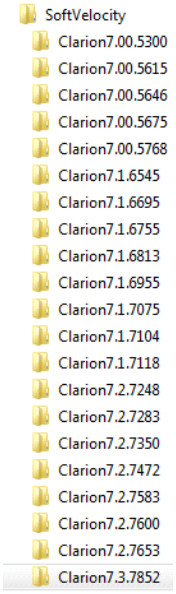


Figure 1. A whole bunch of C7 installs

Some, like Bruce Johnson, advise installing Clarion 7 into the C:\Clarion7 directory (or c:\C7 if you want to save even more typing). That's definitely an option, especially if you only keep one install of Clarion 7 on hand. If you have multiple copies installed then you're going to end up with a lot of root directory entries. I like to keep my root directory as clean as possible, but I also like the convenience of short paths.

There are several reasons I keep multiple C7 installs. For one, if the latest build turns out to

have a critical bug it's a lot less trouble to just run the previous version than it is to uninstall the last version. For another, migrating isn't an all-or-nothing venture. I can have some apps (perhaps with heavy third party dependencies) in an earlier version, and other apps in the latest version.

However many versions of Clarion 7 you have installed, there's another way to alleviate navigation problems besides hard coding root directories: use symbolic directory links.

As of Windows Vista there's a `mklink` utility for just this purpose. At a command prompt type

```
mklink /d NameOfSymbolicLink DirectoryLinkPointsTo
```

For example, if I want to create a C:\C7 link to my C7.3 install from Figure 1, I use the following command (note that quotation marks are required when the path has a space character):

```
mklink /d c:\c7 "C:\Program Files (x86)\SoftVelocity\Clarion7.3.7852"
```

The link appears in Windows Explorer as a shortcut, and functions just the same as any other directory.

The `mklink` utility only creates links, it doesn't provide a means for deleting them.

The easiest way to delete a symbolic link is via Windows Explorer. That removes only the link, not the directory to which it points.

From the command prompt things are a little less intuitive. If you use the `del NameOfSymbolicLink` command Windows will warn you that you're about to delete all files inside that directory. You probably don't want to do that. Instead use the `rmdir` command:

```
rmdir NameOfSymbolicLink
```

That will delete just the link and not the contents of the linked directory.

You can create as many symbolic links as you like. I prefer to collect mine under a directory off the root I simply call "1". And because I frequently work from both my C and D drives I create duplicate "1" directories with identical shortcuts, so whenever I'm at a directory prompt I only need to type \1 and press Enter to see a list of shortcuts. That also helps keep my root directories as clean as possible.

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).

Article comments

by Russell Eggen on December 15 2010 (comment link)

When uninstalling a version of C7 (or anything else installed by SetupBuilder), I simply drag the log file onto the uninstall program. For me, its just more fun than deleting something .

Nice tip! I was not even aware of symbolic links. I plan to use this in the future.

---

by Bob Roos on December 16 2010 (comment link)

Just a few days ago I did something similar in Windows 7 after I was tired of drilling down and drilling down. I made the Clarion install directory a Library. Now it appears in my list of libraries and that is easy to navigate. I think "Libraries" are "mlinks" with a fancy name and a better facility for maintenance.

In the case of multiple installs, they will show up as first level folders under the library. It should still speed up finding them because Win 7 (too many things at 7) gives easy access to libraries.

---

by Mark Sarson on December 16 2010 (comment link)

I've used this before to move some windows directories to a different drive.

I have a client who has MS Small Business Server 2003 on a Dell server. Dell in their infinite wisdom put the OS on what is nowadays quite a small drive (12GB).

Following some instruction I found on a website somewhere (I don't have the link at the moment), I moved all the windows updates to a folder on the clients much larger Raid 5 array, then went into the Windows folder and used mklink to convince windows that the original Windows Update Directory was still on the OS driver.

That freed up around 6GB of space, and all of a sudden the server worked as it should again ;) and what's so cool is that all of the registry entries etc. don't need to be altered.

Thanks for the article Dave.

Regards

Mark Sarson

---

by Dave Harms on December 16 2010 (comment link)

Bob,

Excellent idea. I imagine libraries do use symbolic links at some level. It turns out there's a Windows API to manipulate libraries.

I wonder if SetupBuilder has this functionality. I'll ask.

---

by Dave Harms on December 16 2010 (comment link)

Mark,

Thanks - and that's a nice trick for effectively extending a C drive!

---

by Brahn Partridge on December 20 2010 (comment link)

There are also various GUI options to manage your symlinks. I have used this one before with success - Junction Link Magic

BACK TO TOP

Merry Christmas! ClarionMag's Holiday Schedule And Upcoming Articles

By Dave Harms

Posted December 23 2010

The year is drawing to a close, but it's not quite done yet. We still have a few more articles to come, including:

- Steve Parker's top ten things to like about Clarion 7
- Robert Barton's "Birth of a Compendium", a testament to the power of Clarion and the usefulness of third party products.
- Mark Riffey on what it would take for him to open his wallet and buy Clarion 8.

With a little luck we'll also have a review of NetTalk 5 and an update on the ClarionTest unit testing framework.

The Clarion Magazine office will be closed Friday, December 24th. We'll be back on Monday Dec 27th.

To all who celebrate the season, we wish you a Merry Christmas!

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).

Article comments

BACK TO TOP

ClarionMag Tip of the Week #4: Double Your App's Available Memory On Win64

By Dave Harms

Posted December 24 2010

All 32 bit applications running on Windows are, by default, limited to 2GB of RAM. That's not always enough, as some Clarion developers with really big apps are discovering.

The long term solution is for SoftVelocity to produce a 64 bit compiler, and there's definitely hope this will happen.

In the meantime, you *can* give your 32 bit Clarion 7.1 and later apps access to somewhere in the neighborhood of 4GB of RAM, *if* your application is running under a 64 bit version of Windows. All it takes is one wee little line of code.

In Clarion 7.1 SoftVelocity introduced the LARGE_ADDRESS statement for EXP files. From the help:

> The LARGE_ADDRESS statement
>
> LARGE_ADDRESS
>
> Alternatively, you can add the following to the LNK file:
>
> /LARGE_ADDRESS
>
> With either option set above, the linker adds the IMAGE_FILE_LARGE_ADDRESS_AWARE flag to the PE header. If this flag is not set, the 32-bit application receives 2 GB virtual address space.
>
> With NT architecture, every 32 bit process runs in the 4GB virtual address space, but the amount of the virtual memory available for process data and code is dependent on Windows characteristics and settings in the process main executable's PE header.
>
> WOW64 enables 32-bit applications to take advantage of the 64-bit kernel. Therefore, 32-bit applications can use a larger number of kernel handles and window handles. However, 32-bit applications may not be able to create as many threads under WOW64 as they can on x86. On some processors, there is less virtual address space available, and each thread contains a 64-bit stack (usually 512K). On the x64 processor, each 32-bit application receives 4 GB virtual address space in the WOW64 environment, if the application has the IMAGE_FILE_LARGE_ADDRESS_AWARE flag set in the image header. If this flag is not set, the 32-bit application receives 2 GB virtual address space.

The EXP option is easy to implement. Go to your application's global embeds, and simply add the text

LARGE_ADDRESS

to the Inside the export list embed point (Figure 1).

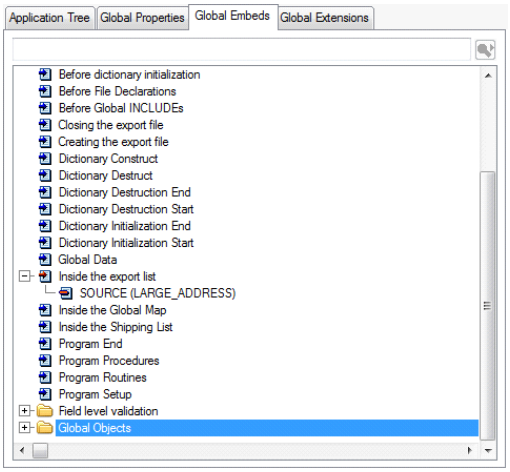


Figure 1. Enabling 4GB of memory access

Generate and compile your application and it's good to go!

You can confirm the flag has been set if you have Microsoft's dumpbin utility (which comes with Visual Studio).

On my machine I used the following command to verify the header. You may need to adjust the dumpbin path portion, and you will definitely need to point it at your EXE and not the path shown.

```
"C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\bin\dumpbin" /header s
  "d:\dev\c7\people\people.exe"
```

Here's a portion of what dumpbin returned:

```
PE signature found

File Type: EXECUTABLE IMAGE

FILE HEADER VALUES
   14C machine (x86)
   7 number of sections
   4CFF850B time date stamp Wed Dec 08 07:15:55 2010
   0 file pointer to symbol table
   0 number of symbols
   E0 size of optional header
   81AE characteristics
   Executable
   Line numbers stripped
   Symbols stripped
   Application can handle large (>2GB) addresses
   Bytes reversed
   32 bit word machine
```

Note the third-last line: `Application can handle large (>2GB) addresses`. As noted above, this only applies when your app is running under 64 bit Windows.

Although the Clarion help doesn't specifically say whether your modified 32 bit EXE will

still run on 32 bit Windows, I don't see any reason why it wouldn't; in my simple test of the People app the EXE runs fine on both 32 bit and 64 bit versions of Windows 7. The only difference is that on 32 bit Windows the EXE will only get up to 2GB of RAM.

The long term solution to the 2GB memory constraint is a 64 bit Clarion compiler with (presumably) up to eight terabytes of addressable memory. In the short term, another 2GB of memory probably won't hurt.

If you're running into memory constraints, give this option a try. And please post a comment with your experiences.

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).

Article comments

by John Hickey on January 1 2011 (comment link)

Brilliant! I tried it in C7 and it didn't work, I kept poking around, finally realized I was still compiling in C6 mode! So ok, I have to compile in C7 mode, but this is a great tip!

---

by Dave Harms on January 3 2011 (comment link)

John, I think you can still make C6 apps work this way but you have to manipulate the EXE directly using the editbin utility that comes with Visual Studio:

http://www.point41.com/index.php/tag/editbin/

It should be possible to set that up as a post-build task.

BACK TO TOP

The Best Thing About Clarion 7

By Steven Parker

Posted December 27 2010

We've all heard the reasons why Clarion 7 is supposed to be better than any previous version of Clarion. I won't repeat them here. I will tell you that I found only of them two personally motivating: (1) the size limits on embeds and complex structures are eliminated and (2) it can run natively in all modern versions of Windows.

In 16 bit Clarion IDEs, embeds are limited to 16k. If you exceed this limit, you won't know about it until you try to save your work. At that point, you will get a warning and the embed will be truncated. Your work will be lost and, even if you cancel all the way out of the app, there is nothing you can do to get it back.

I discovered this the hard way. After I had lost code a couple of times, I took to block copying new code before trying to save (I was almost always adding new code at the end of embeds, so this trick worked pretty well for me). Eventually, given my coding style, I simply began a new embed when I hit 1500 lines in my current embed.

Along with the limit on embeds in the 16 bit Clarion IDE, there is a limit, 64k I believe, on window and report structures. Developers of really complex windows have no choice but to create controls at run time or, laboriously, remove controls and/or shorten FEQs and/or other Labels (and the later is only a short term solution). Additional report bands, bands that exceed the limit of the Report Formatter, have to be created in a Data embed. And because the Report Engine does not know about them, these bands must be printed manually.

Run time controls in windows are difficult to maintain. They certainly cannot be maintained visually in the Window Formatter.

Perhaps the ability to run Clarion in 64 bit O/S's is more important to me than it is to many others. The 32 bit IDE gives me the option to run C7 on my desktop or in a virtual machine. On a 64 bit O/S, the 16 bit Clarion IDE *must* be run in a VM. C7 gives me the choice. C5/5.5/6 doesn't. It is not that I am averse to VMs, it's that I like having options.

The Best Feature of C7

I recently found a feature of C7 that beats anything SoftVelocity or anyone else has put forward in favor of C7. I was just plain gobsmacked when I realized I had found something

in C7 that improved my code and did so virtually effortlessly.Bar none, the best feature of C7 is its compiler.

To understand my new found enthusiasm for this sterling feature, you need to understand two things. First, I have always relied on the compiler to catch my fat finger mistakes, at least the more egregious of them. Second, I have spent the last several months doing nothing but up-convert apps from C5.5 and C6.

If I type something like:

```
?List1{PROP:SelStart}  - RECORDS(ItemQueue)
```

instead of

```
?List1{PROP:SelStart}  = RECORDS(ItemQueue)
```

I expect the compiler to complain. C6 and its predecessors like that code just fine. C7 doesn't.

I actually did type a line with a minus sign where there should have been an equal sign (or was it the other way ‘round?). It was months before customer bug reports started coming in and got me looking at my code.

Here are some examples from my recent emersion in old code. Before beginning my “Top 10”, this is the time to go get a beer, put your feet and get ready enjoy “Are you smarter than a 5th grader, Clarion style?”

My Top 10

(I have taken the liberty of changing actual variable and class names in the code snippets shown below. Except for that, all code is from real, compiling 5.5 and 6.x apps.)

My top 10 …weird-nesses, not in any special order, are:

Number 1:. An earlier version of Clarion let me include a third party global extension and then call methods/procedures from one if its Procedure Extensions without actually populating the Procedure Extension on a procedure. Uh, yeah, the method/procedure is in the Procedure Extension, not the global extension. I can’t explain it; I don’t understand how it compiled before but I saw this twice.

Number 2: I found this in a compiling app:

```
?   IF myVariable = True
?     cMyClass.myMethod('myLiteral: ' & my:Variable)
    END !  if
```

See the mismatched `END`, outside the debug structure?

Number 3: I found prototypes like `(dataType Label)`. The app called the procedure with `(dataTypeXX Label)` or `(dataType YYLabel)`. I found a lot of examples of this when passing named Groups and Queues where the actual instantiation of the Group or Queue has a slightly different string in the `dataType`. (Is this a case of the compiler doing what I meant and not what I typed?)

Number 4: 5.5 didn't seem to mind:

```
IF Request = Delete Record
```

See the space? I saw this twice.

Number 5: One that 7.2 also features: a LIB or DLL can be included as External module more than once. The C7 compiler will error, though the IDE allows it. Earlier compilers compile the code.

Fixing this was a real adventure.

Number 6: `DefineListboxStyle` throws "Routine not defined;" this showed up a lot. But the routine is there, it is generated, in the CLW. This happened a lot when converting apps using a specific third party template. (There were other routines, from third party templates, that did this also but `DefineListboxStyle` did it quite frequently, quite, so I pick on `DefineListboxStyle` here.)

Number 7: I found an object instantiation:

```
myObject cClass1
```

in a global data embed. I also found another object instantiation to the same name. (The compiler knew what the developer was talking about, eh? It really is a duplicate.)

```
myObject cClass2
```

elsewhere in Global Data. In other words, the earlier version allowed creating two objects with the same Label.

Number 8: Queues – with the same Label -- could be declared both globally and locally without complaint (don't know if there was a warning or not in the original, but C7 sure gets upset). Yup, two queues, two scopes, one Label, no error.

Number 9: Class files can have a procedure declared in the INC but no code implementing the procedure in the CLW.

By itself, that's not wrong.

But I did find the procedure defined in another class in the same CLW. And, no, that method was not declared in the (second) class definition in the INC (both classes were defined in a single INC).

Calling the method actually worked!

Number 10: `COMPILE('endXYZ', MyTerminator = 1_`

5.5 allowed this. No end parenthesis and it compiled. Clearly, the earlier compiler had more intelligence or was more helpful. Maybe it could read my mind?

"Surely you jest!"

No, I'm not making these things up. Each and every one comes from a real app, some mine, some not. All I've done is disguise Labels. Eight of the ten errors occurred multiple times; the two that didn't (numbers 2 and 10) were so outstanding, I just had to include them. I have more in my notes (converting 200 apps, one after another, non-stop for a few months provides a lot of fodder).

Clarion 7 caught each and every one. That's how I know about them, of course.

Individually, some are kind of funny. You can certainly imagine sitting there looking at them, wondering "just how the [expletive deleted] did 5.5/6 compile *this* [expletive deleted] thing?"

Collectively, they're mind boggling. I can only imagine all the bugs people have been chasing – I certainly had that … opportunity – when the compiler could have (and should have) thrown an error.

The C7 compiler isn't perfect. For example, I created a small app in C7 and created a local variable in its one procedure. Accidentally, I ticked "Reference." The first time I hit an embed containing the variable, GPF! You'd think the compiler would tell me about an unitialized reference variable being used.

Similarly, though this is more an IDE than a compiler issue, clicking on an error in the Error Pad is not 100% certain to take me to the actual error. I found this fairly frequently when an error was in an `INCLUDE('myFile')` statement in a global embed. Sometimes I'd find myself in the generated code instead of the Embed Editor (fixing a problem in generated code is a waste of effort; on next generation, changes will be overwritten; but, at least, I can locate the procedure and approximately where in the procedure the error is located).

With a decent text search tool, locating where a particular string occurs – the Error Pad is very good at showing me the offending string – almost always makes finding and correcting the problem "do-able."

Conclusion

I make mistakes. I forget `END`s, I reverse letters (yes, I, even I, have lysdexic moments), I mispel variable names and misstipe expressions.

The compiler's job is to catch these mistakes. I remember cases where the compiler

disallowed things that the language manuals say are legal; I remember commenting you can't win an argument with a compiler. So why can't it have the courtesy to catch duplicate object names or invalid `OMIT` scopes?

I don't really expect the compiler to be perfect. Nothing created by human hands can be.

Okay, I *do* expect it. (On the other hand, I'm not willing to hold my breath until it is perfect.) But the evidence is clear. The 5.5 and 6.x compilers passed quite a bit that is just unforgivably wrong.

Being able to run in any O/S I choose is nice. Not having my embedded code truncated is *very* nice. Having confidence in my compiler … priceless.

Steve Parker started his professional life as a Philosopher but now tries to imitate a Clarion developer. He has been attempting to subdue Clarion since version 2007 (DOS, that is). He reports that, so far, Clarion is winning. Steve has been writing about Clarion since 1993.

Article comments

by Djordje Radovanovic on December 27 2010 (comment link)

Another GEM

I tried to convert some of my major application and compiler show me an error I have never seen before in CW6.

I declared variable as

var[4,61]

and in program code using even var[11,10] what obviously I am not suppose to do. CW6 silently swallow this error but CW7 has much more strict compiler and I like it. Who knows how many errors did I have in my applications which I couldn't explain with this CW6 forgiving compiler?

---

by Vince Sorensen on December 31 2010 (comment link)

The funny thing is that Clarion 6 will detect duplicate groups in the templates while Clarion 7 doesn't...

---

by Jane Fleming on December 31 2010 (comment link)

Steve,

Although I haven't used C7 much, I've been tripped up **numerous** times by the inadvertently-ticked "reference" check box. I swear I'm not doing it! It's cosmic rays swinging around Titan and manipulating my mouse driver while I'm not looking!!

I haven't done it quite enough to have that be my first search item when I suddenly encounter a runtime GPF. But I mouth increasingly deprecative comments each time I do this...

Jane

---

by Devan Sabaratnam on December 31 2010 (comment link)

Great article Steve. I wasn't aware that the C6 compiler let so much through...perhaps I am a great coder??? Nah! :)

I concur with Jane though - the 'phantom reference ticker' once caused me to lose two days trying to find why my app was crashing. Happened so many times since then that now my first step in fixing a GPF in my app is to go through checking for a ticked 'Reference' box in my Global and Local variable declarations...

---

by Jimmy on January 1 2011 (comment link)

Steve, I program to get things done in the quickest and simplest ways (Been with clarion since version 2008 dos 1987 The most amazing thing in Clarion 7 is the ability to copy-and-paste virtually anything making programming a create once use many times wherever you want - Just amazing...

---

by Devan Sabaratnam on January 1 2011 (comment link)

Oh yes, Jimmy's response reminded me how much I love, I mean Love..that is LOVE, the way that C7's IDE allows you to copy complex structures such as QUEUES and GROUPS from procedure to procedure in the App generator without having to do it field by field like before!

I've had to move queues containing about 30 fields from the Global space to various Local procedures a few times during my last project and this ability was a godsend. Especially how we can paste the queue into a text editor, tweak it a little, then recopy and paste it as an entire structure in the C7 IDE.

Correct me if I am wrong, but I believe you could not do this in C6...

---

by Adriaan van Ieperen on January 3 2011 (comment link)

It is possible to copy local data in C6. But it isn't as easy as it is in C7. You have to open the procedure properties and click on the "..." button next to data. That will give you a source view of the local variables. From there you can copy/change/paste.

BACK TO TOP

The Birth Of A Compendium

By Robert Barton

Posted December 28 2010

A compendium can be defined as ‘a concise, yet comprehensive compilation of a body of knowledge’.

This particular Clarion-powered compendium came into being after a phone call from a Customer who wanted a means to gather all the information they had on each of their customers.

The background

I had just finished a project to digitise my own ‘bits of paper’ holding tips and tricks for clarion and other software. These had grown over the years to well over a hundred pages of notes – of course they were all well indexed etc!

I had purchased ProImage and ProScan sometime before as products too good to miss but had not had a proper use for them. So the ‘bits of paper’ project was an ideal way to put them to the test. This was easily accomplished – it just took a couple of days to create a basic but effective app that enabled me to scan the paper notes and store them to disk with a database entry for each note. I added a little bit of info to the database so I could search for key words to locate a particular note or notes.

Then came the phone call….

The customer requirement

The Customer already had a comprehensive system covering the sales order processing, service call management, telephone billing and other miscellaneous tasks. We had implemented outgoing correspondence to that app using Word and storing the documents by Customer.

So some of the requirements were already known but we came up with the following list to store on the computer, if possible:

1. Incoming and outgoing mail both paper and electronic.
2. Sales information including quotations, invoices, service agreements and other relevant

documentation (e.g. scanned manuals/product documentation relevant to the customer)
3. The customer database already had notes of sales calls, call backs and service calls. These would not be included in the compendium.

My client's reply when I suggested recording telephone calls in the compendium was unprintable, and he is in the telecoms business!

The ToDo list or design

Out came the Clarion toolbox! On second thought a design for the project was a better bet!

Could we use the previous 'bits of paper' app as a basis for the new compendium? The answer was 'YES'. Great – reuse where possible! Of course the scope for the compendium was much larger and the ructions it would cause in the overall system were considerable.

The first decision was to store the content in the compendium by itself in its native format – that format being PDF, DOC, DOCX, TIFF, HTML etc. That meant we only had to create a single file to store this information and any other information we needed to help search for records that were stored – thus the Compendium was named and created.

We created fields for description and keywords and the file had indices based on customer id, keywords, description to make retrieval as fast as possible. In addition to make day to day retrieval easier, we set up keys for accounting periods and specifically for customer invoices.

The entry to the compendium would be by the way of the standard Clarion browse/form paradigm, and the stored information (at the top level) would be visible from the browse. Opening the form would gives access to the stored data with the relevant content being opened in a thumbnail before the full content needed to be opened. If the full content was opened then more options were to be available – e.g. in the case of scans access to the scan module to be able to manipulate the images etc.

That was the easy bit – now we needed to decide how to capture the content for the compendium.

In my toolkit I had the capability to scan images, to access Outlook (the customer's mail handler) and extract the email information, create PDFs for invoices, create Word documents and display the retrieved content. So all was well!

Some work on the original system would be needed to produce the invoice documentation for instance and to integrate the compendium into that application. Nothing difficult!

The actual compendium

So how was it done? I decided to create a standalone version of the compendium app as I was not sure how it was going to happen. I did not want to incorporate this new functionality

into the main app which was made up of a number of DLLs etc. until I had proven the results from the standalone version.

I created the browse/form and installed the software from ProScan onto the form. The browse was enhanced using Clarion Handy Tools to give effective search and query methods. I checked it out to be sure it was working. No problem there – I did a scan after setting the needed parameters for the scanner I was using. Once again it was simple!

As can be seen from Figure 1 the form displays a snapshot of the scan. This is part of the ProScan templates.

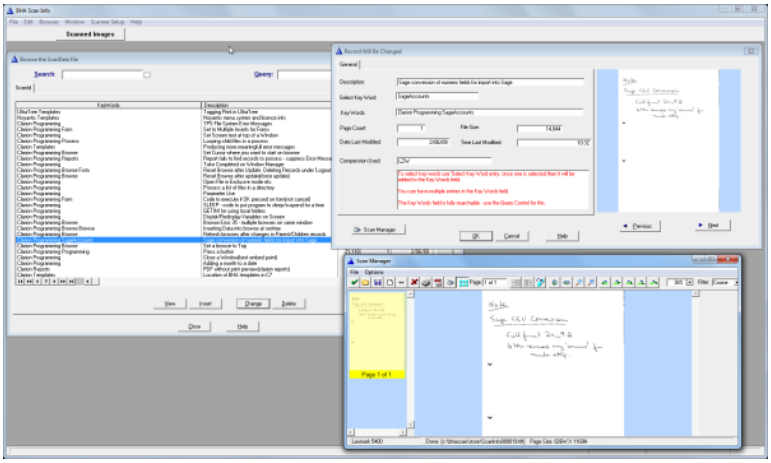


Figure 1. The scanning form (view full size image)

However I also needed to be able to produce a snapshot like this for other file types – PDF, doc, html etc. I used File Explorer from Capesoft to do this. When I first started to look into displaying a number of formats, my heart sank at the complexity but once I started implementing File Explorer a whole area of uncertainty disappeared – not to say the amount of time to implement as well. I set up another tab on the form to handle File Explorer and auto switched between tabs depending on the type of file being used (Figure 2). Note File Explorer uses the associations set up by Windows to display the file in its correct output format – very handy!

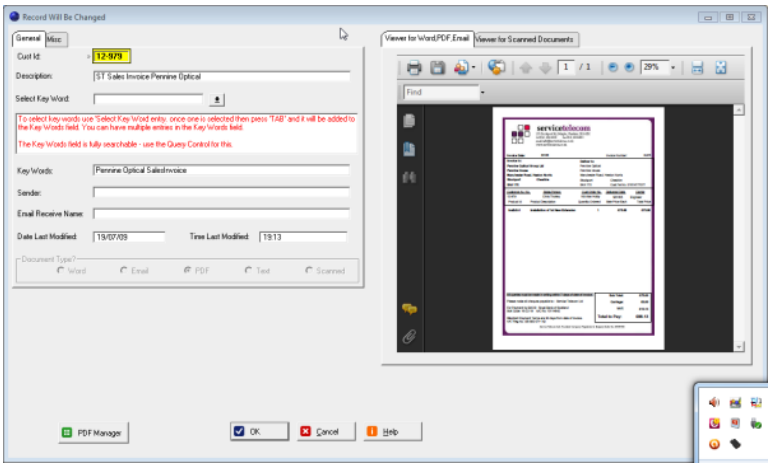


Figure 2. The input form (view full size image)

The PDF Manager button (bottom left) gives a full screen to the PDF with other facilities available. Similar buttons for other document types appear when those types are in use – I retrieve the document type from the compendium file record so that the form can display the correct choice. Remember all this is standard out of the box File Explorer functionality.

Figure 3 shows the tab detail on the final form screen.

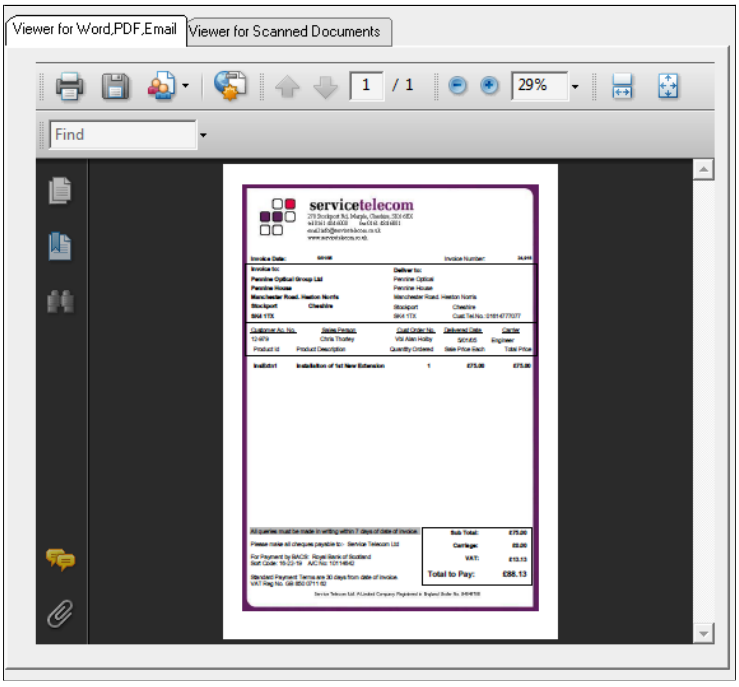


Figure 3. Tab detail

The production of PDFs from Invoices took additional work. Allied to this was the decision to drop the former invoice reprint facility and use the PDF as the invoice reprint. All invoices (there are five different types of invoice in the system) were converted to produce PDFs and these files were stored on disc for retrieval from the Compendium as needed. In the case of the Telephone Billing module the PDFs already had their destination folders so

this was left intact, but the PDFs were linked to the Compendium so that they could be reproduced as needed.

The facilities added so far cover the needs of all the documents wanted for the compendium with the exception of emails, both incoming and outgoing. This seemed to me to be the most difficult request to solve as the Customer used Outlook for this. Outlook to me was at that time virtually unknown – yes I had a copy but had never used it in anger!

In the event, sorting out access to the emails took by far the largest amount of time to implement. This was compounded by my lack of knowledge of Outlook and consequently my understanding of how to get the information I needed out of Outlook. I chose Office Inside from Capesoft to do this. I will not say I found it easy (after all I am an extremely reluctant coder) but with a little bit of help from the team at Capesoft and more time than I wanted reading their documentation I made steady progress.

I consulted with my customer at this point to determine at what level he wanted to retrieve the emails from Outlook. His decision (for which I was grateful) was for him to select the emails he required for the compendium – both incoming and outgoing. His main point for this was there was so much junk involved that he only required a subset of the emails in the compendium, so he would choose the ones that he required! I provided a Clarion process for him to use to extract the emails he required into the compendium (Figure 4).

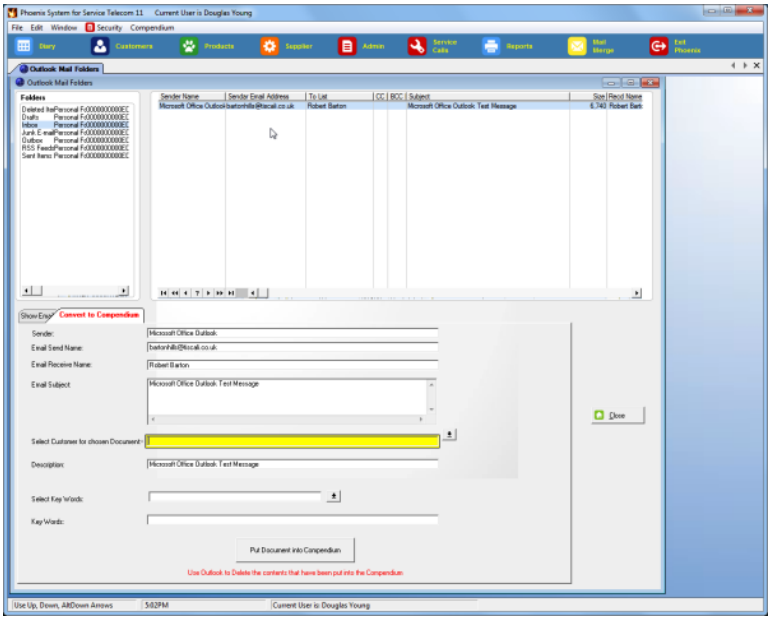


Figure 4. Outlook integration (view full size image)

This process is not exactly suited to a high volume of emails, but it has worked for the customer so far. They have been using the system for approximately eighteen months.

At this point the Compendium application was finished as far as gathering, maintaining, and outputting the content was concerned.

However, I added similar processes to handle other types of content so that the Customer could add ad hoc content to the compendium as needed.

In addition, I created various conversion routines to archive content. There were PDFs in the form of several years' worth of telephone bills that needed this treatment.

At this point a meeting with the Customer took place to do a full trial of all the routines in the Compendium. Various suggestions were made to improve the usability, which were implemented – mainly moving fields on the forms.

Up to this point the application had taken five working days (long ones!). The next task was to fit the content of the app into the main, multi-DLL application. This took more time to integrate than I had hoped, but was accomplished in another couple of days.

The last task was to log all the additions to the database – this was simple to do, but was tedious as there were a number of places where the content was created and therefore the compendium record had to be created.

Prior to delivery, it took me a full day of computer processing to set up the initial compendium from the historic data which in certain cases went back more than seven years.

So finally after two working weeks, the app was delivered to the Customer. Feedback was positive and since then there have been very few problems reported and the features provided have met with their needs.

A caveat

This application, despite all its functionality, is really just an application suitable for a known user. It would take a considerable time to make it suitable for multiple companies to use. This is always the case when you move from a single Customer to a more general market. Is all the functionality that you could want included – certainly not!

I make no apologies about this, but feel it should be stated!

Postscript

I used many other tools in this system to provide a nice looking, user friendly application. But the main achievement was the time it took to put together – less than two working weeks.

If I had been asked to do this from scratch, I would have refused as it would have been well beyond my software capabilities, but thanks to the Clarion toolbox it was possible to do and to do it quickly.

There is a cost to developing an application this way, I have a large set of third party tools that I use all the time, with a value of over US$ 2000.00. If I had had to buy the ones used for this project, the cost would have been US$1200.00. An investment of that size plus two

weeks of work yielded a very cost-effective set of the features for my customer

Warning to non-Clarion users

This is not an article on how good Barton Hills Associates are, but an illustration of how productive you can be using Clarion and its associated tools to satisfy a business need – which is the *reason* we are in business and why Clarion is our main toolbox provider.

The breadth of the Clarion toolbox is one major reason why we small groups of developers can still compete today despite off-shoring and it's like – the other part being that we are close to our customers, know their needs, and can respond with timely results.

Article comments

by Alex McCullie on January 1 2011 (comment link)

Add-ons are very attractive to a developer - instant functionality for relatively a small cost. I've used them a lot in the past. However they have a dark-side, particularly for creating apps designed to be different from out-of-the-box Clarion applications. Here are some caveats that plagued me over the years: (1) bug resolutions - who is responsible Clarion itself or one of the 3rd-party products plugged into the problem app?; (2) Upgrade maintenance - Clarion and each 3rd part add-on will have a string of (sometimes incompatible) version updates and upgrades, free and payable; (3) incompatible interfaces - as you move away from a pure Clarion look & feel the integration of add-ons becomes more problematic; (4) unwanted dependency - the Clarion market is small and so one add-on market is typically very small. If the developer decides to give it away and you one or more commercial apps dependent on that add-on ouch! even with source available. I still remember the heartache for PowerBrowse users with the move to ABC.

I'm not advocating no add-ons rather prudence in a potential candy shop. Alex

---

by Robert Barton on January 1 2011 (comment link)

You are, of course, absolutely right! It is difficult to be sure you have a winner when you buy software tools.

However it has to be remembered that Clarion is also a tool ans is subject to the same problems of market forces. Many of us who are still Clarion users must be pleased to be able to still use the product despite its trials and tribulations.

I have been caught out in the past - PowerBrowse comes especially to mind. But then again I have found other 3rd party suppliers have provided excellent products with an upgrade path that matched or bettered my needs, in particular Capesoft and Handy

Tools to mention just two suppliers.

Given the problems with new versions of Clarion, timeliness in particular,I've tried to remain behind until my needs are available. This has usually avoided most of the heartache!

In the case of Clarion7, I had to move before I wished due to operational problems with the C6 ide. On balance I wish I hadn't had to do that. I am still especially plagued by the new Clarion Report Writer which has suffered neglect by SV, such that I have major workarounds still with that product and this isn't a 3rd party product! Well not entirely, its SV official choice.

Despite these problems I have been able to use Clarion profitably for over 20 years and that I feel is a major achievement for Clarion and its associated 3rd party products. Given all the software upheaval over that timescale, it make the Clarion paradigm a survivor in an uncertain world. Something we users need to remember with gratitude, as many other products have been and gone with all the pain that involves. Clarion as a pain free experience? No more like dental visits than major surgery! I can cope with that!

BACK TO TOP

ClarionMag Tip Of The Week #5: Zoom In On Small Prompts

By Dave Harms

Posted December 31 2010

This week's tip comes from Bruce Johnson, who points out the oft-overlooked "zoom" feature in the AppGen.

Any time you're on a template prompt and you can't quite see all the information, just press F10.

Figure 1 shows a procedure's short description exceeding the displayable area.

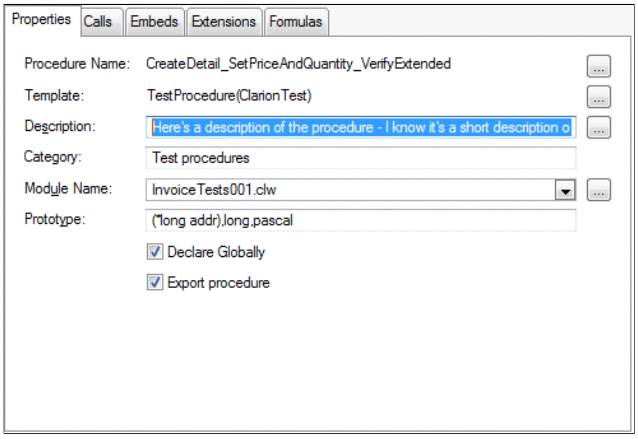


Figure 1. Procedure description text

In Figure 2 I've pressed F10 to bring up the zoom window.

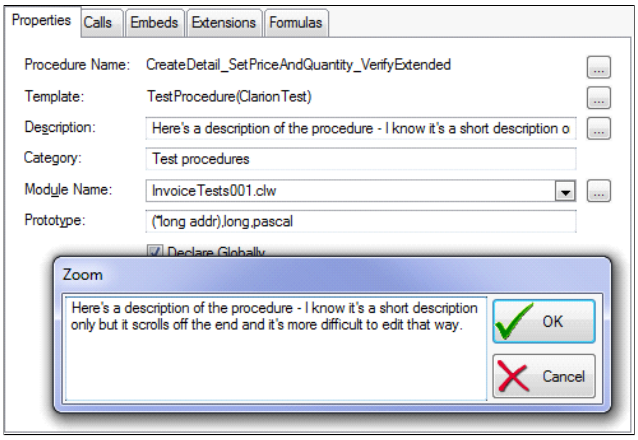


Figure 2. The zoom window

Procedure description fields are actually pretty generous - there are lots of wee little ones all over the templates where you often can't see all the text.

I've been using F10 a lot lately on procedure renaming. My unit test procedures tend to have some fairly long names that just don't fit on the procedure rename dialog (Figure 3).

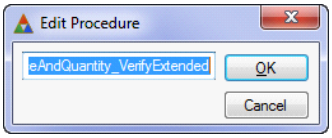


Figure 3. Renaming a procedure

Pressing F10 instantly makes the procedure name readable.

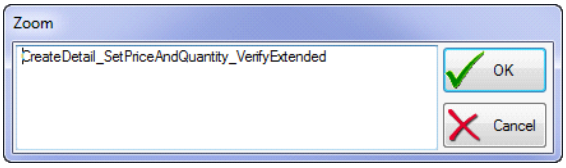


Figure 4. Zooming the procedure name field

The next time you're puzzling over a too-small entry field, just press F10!

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).

Article comments

BACK TO TOP

Start Testing Your Clarion Win32 Code With ClarionTest

By Dave Harms

Posted December 31 2010

A year ago I wrote a series titled "The Problem with Embeds" in which I argued that Clarion's system of embeds, while powerful, fosters some really bad coding habits. As Clarion devs we tend to throw all our code into embeds which makes that code almost impossible to reuse and to test.

Ah, testing. How do you test your Clarion applications? If you're like most Clarion devs, you test code by sitting at a screen, typing test, clicking buttons, and looking for results. Or, worse, you let your customers do your testing for you.

This kind of testing is slow, unreliable, difficult to measure and difficult to repeat with accuracy. Yes, a certain amount of UI testing is essential, and that kind of testing is by its very nature hard to do.

But most of the code that we write isn't UI code; it's business logic. It's code that does stuff with data. And *that* we can and should test.

Unit testing with ClarionTest 0.3

In Part Four of that article series from a year ago I introduced a unit testing framework for Clarion Win32 programming (originally called CTest). I've updated that framework several times since (it's now called ClarionTest), and this article covers the 0.3 release.

The framework consists of an executable program, a template, and some accompanying source code.

You create your unit tests as procedures inside a DLL. The only purpose of that DLL is to be a home for your tests, and you can create as many different test DLLs as are necessary. Register ClarionTest.tpl and be sure to add the global extension to the DLL.

When you create new test procedures in the DLL, do so using the Test Procedure from the Defaults tab (Figure 1).

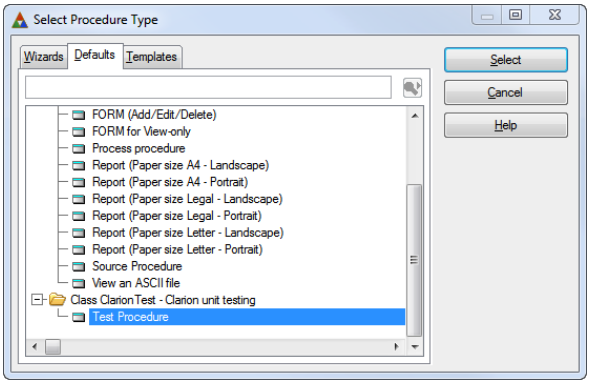


Figure 1. Creating a test procedure

Be sure you use the Test Procedure from the Defaults tab and not the Templates tab. See The Problem With Embeds, Part 5: Unit Testing The InvoiceDetail Class for more on creating test procedures (but be aware that the generated code has changed significantly, as descibed below).

You run the tests by building the test DLL, and then telling ClarionTest.exe to load up the DLL and execute its tests. Figure 2 shows ClarionTest in action.

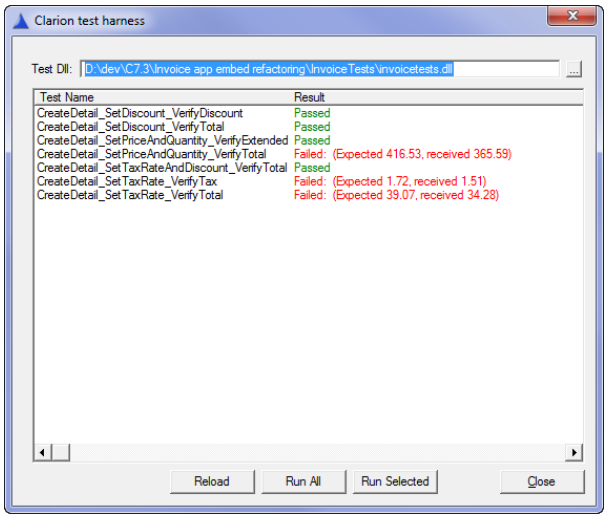


Figure 2. Running some tests

The individual tests are typically quite short, because they call either functions or (preferably) class methods. In Figure 2 I'm testing a class that embodies the logic needed for the Invoice app that ships with Clarion. Figure 3 shows the source code for the `CreateDetail_SetTaxRateAndDiscount_VerifyTotal` method in the embeditor view.

```
CreateDetail_SetTaxRateAndDiscount_VerifyTotal PROCEDURE  (*long addr) ! D
! Start of "Data Section"
! [Priority 3500]

! [Priority 8500]
detail                                                  InvoiceDetail

! End of "Data Section"
  CODE
! Start of "Processed Code"
! [Priority 50]

    addr = address(UnitTestResult)
    BeginUnitTest('CreateDetail_SetTaxRateAndDiscount_VerifyTotal')
    !---------------------------------------------------------------------
    ! Write your code to test for a result, using the AssertThat syntax.
    ! At present there are two different assertions you can use, IsEqualto
    ! and IsNotEqualTo. You can pass in any data type that Clarion can
    ! automatically convert to a string.
    !
    !   AssertThat('a',IsEqualTo('a'),'this is an optional extra message')
    !   AssertThat(1,IsNotEqualTo(2))
    !
    ! As soon as an Assert statement fails there remaining tests will
    ! not be executed.
    !---------------------------------------------------------------------

! [Priority 5000]
    detail.Init(12.17,6)
    detail.SetDiscountRate(12.2)
    detail.SetTaxRate(7.9)
    AssertThat(detail.GetExtended(),IsEqualTo(73.02))
    AssertThat(detail.GetDiscount(),IsEqualTo(8.91))
    AssertThat(detail.GetTax(),IsEqualTo(5.06))
    AssertThat(detail.GetTotal(),IsEqualTo(69.17))

    return 0
! [Priority 9950]

! End of "Processed Code"
! Start of "Procedure Routines"
! [Priority 3500]

! End of "Procedure Routines"
```

Figure 3. A test method

As Figure 3 shows, test procedures are just source procedures with a special prototype and a few extra lines of auto-generated code (both supplied by a procedure template).

In Figure 3 there's one declaration - an instance of the `InvoiceDetail` class. The code section executes some methods on the `detail` object, then tests a number of conditions using the `AssertThat` function.

Release 0.3 changes

`AssertThat` is a major change from the first release of ClarionTest, which required you to write a bit of logic along with your test code. `AssertThat` is patterned after NUnit's `Assert.That` syntax, although this Win32 implementation is inevitably a bit crude by comparison.

`AssertThat` greatly simplifies the code. The syntax is typically either

```
AssertThat(result_of_some_action,IsEqualTo(expected_value),<optional message>)
```

or

```
AssertThat(result_of_some_action,IsNotEqualTo(expected_value),<optional message>
```

Because of Clarion's automatic data conversion, the `IsEqualTo` and `IsNotEqualTo` functions handle the vast majority of test situations (although I'm thinking of adding `IsGreaterThan` and `IsLessThan`).

Another small change from earlier releases is that you no longer need to prefix your test function names with `Test_` - you're free to use any function names you like. A side effect of this change is that test procedures are now loaded in their original case. Prior to version 0.3

the procedures were loaded from the DLL's export table and were in all caps until they had been called for the first time.

Dealing with failures one at a time

In general, unit tests should be very small and very fast, and they should only test one thing. In Figure 2 I have a number of `AssertThat` statements which suggests that I'm testing more than one thing, and it might be better to split these up into separate tests. But sometimes it's convenient to issue a series of `AssertThat` calls; just keep in mind that only the first `AssertThat` failure within any one test is reported. So if you have three `AssertThat` calls, and the second and third are failing, you'll be notified of the second call's failure. Only after you fix that problem will you see the message from the third call's failure.

Calling ClarionTest conveniently

You may find it helpful to have ClarionTest on the C7 Tools menu, for quick access. Choose Tools | Options, and from the list of Options choose Tools again. Click on Add to create an new external tool entry. Set the Command field to point to ClarionTest.exe, and set the Arguments field to `"${TargetPath}" /run` as in Figure 4. Quotation marks around `${TargetPath}` are optional but you'll need them if you have any spaces in your path names. You'd think you'd also need them in the Command field if there are spaces in the path, but that isn't the case.

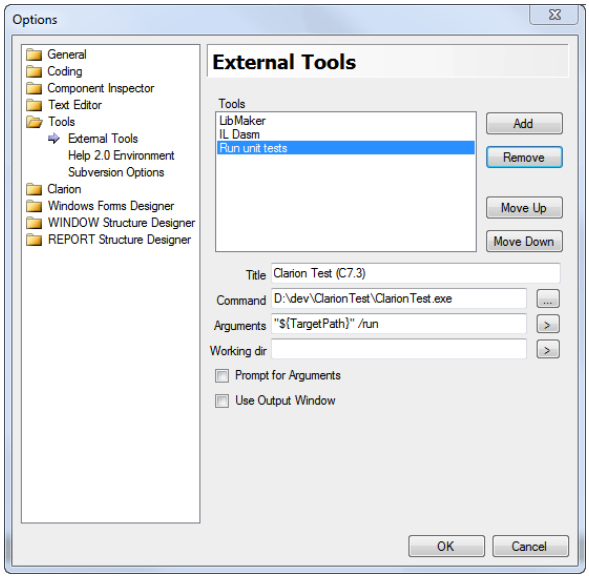


Figure 4. Setting up CTest as an external tool

With the test DLL as the active window in the IDE, choose Tools | Run unit tests. ClarionTest should load the DLL and execute all the tests. If you don't want the tests to execute automatically just remove `/run` from the command arguments.

Calling ClarionTest automatically

As convenient as the Tools menu is, I've come to prefer having ClarionTest run automatically whenever I do a successful build of the test DLL.

Some of the time I'm making changes to test procedures in the DLL APP, in that case the workflow looks like this:

1. Modify the procedure
2. Save the procedure changes
3. Click on Generate and Make
4. If the build succeeds, ClarionTest pops up and runs my tests
5. I check the results, press Esc to close ClarionTest, and go to step 1.

Most of the time, however, I'm busy making changes to classes, and I may not even have the test DLL's APP open. In those cases the workflow looks like this:

1. Modify some source code
2. Press F8 to build the DLL
3. If the build succeeds, ClarionTest pops up and runs my tests
4. I check the results, press Esc to close ClarionTest, and go to step 1.

To do this just set up ClarionTest as a post-build task. Right-click on the project entry in the Solution Explorer (see Figure 5) and then select Properties.

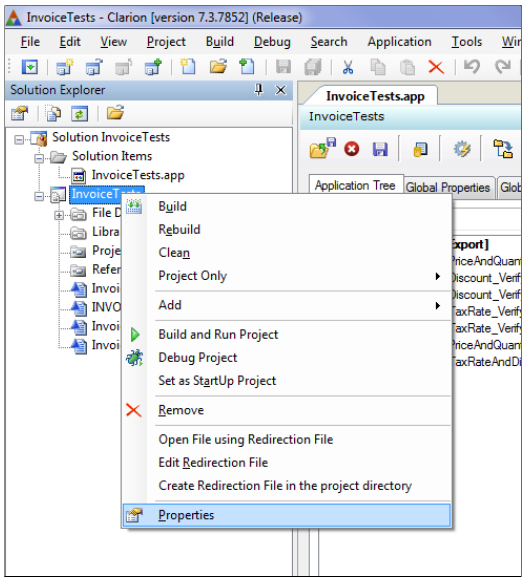


Figure 5. Opening the project properties window

Click on the Build Events tab (Figure 6). You can use the [...] button to look up ClarionTest.exe. Here again, if you have spaces in the command line you'll need to enclose the path in quotes. Add the name of the DLL and, if you want the tests executed automatically, the `/run` parameter.

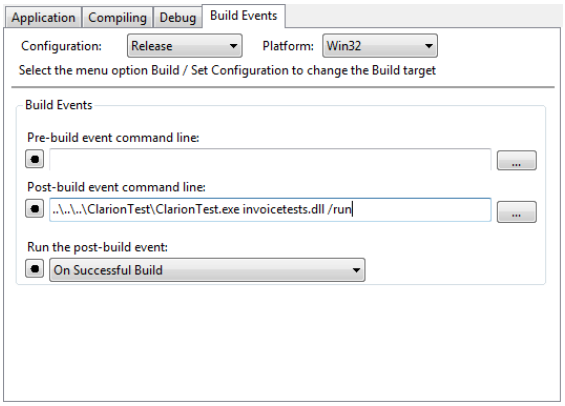


Figure 6. Setting a post-build event

Since my tests are going to run on each successful build, I want them to be fast. Ideally they should run in a couple of seconds or less. I do have a few tests that take longer than that, but I tend to isolate those in their own APP file so they don't impact my other work.

I've spent entire days working in this code/test cycle, creating new tests, refactoring code, finding and fixing bugs. In fact, the most enjoyable way I've found to use ClarionTest is to enable *class design*.

Class design and test driven development

Back when I first started writing object-oriented code, I often found the most difficult part of the job was imagining what the class would look like. Which methods should the class have? What properties? Would I need one class, or multiple classes?

And at some point I realized that it was a whole lot easier if I approached class design, not from the perspective of what I wanted to include in the class (the methods, properties), but *how I wanted to use that class*. So I'd imagine some code that called the class. And that often made the class design task much, much easier.

And then I came across Test Driven Development, or TDD. And it all looked very familiar to me, because in a way that's what I'd been trying to with my "how will I use this class" visualizations.

The crucial aspect of TDD is to begin with code that will not even compile. In my `InvoiceDetail` class test above, I might start with the following test code:

```
detail.Init(11.45,3)
detail.SetDiscountRate(11)
AssertThat(detail.GetDiscount(),IsEqualto(3.78))
```

Since this is the very first code I write, it's just not going to compile. And that's okay. In fact, it's an essential part of doing TDD. I'm using the test as my *starting point*. I haven't written the class yet. So the next step will be to add a declaration for the detail object and a stub class. At that point my code will compile, but of course the test will fail because I won't have any logic in the class yet. *It's only once I have a failing test that I actually start writing the class code.* In TDD, you always begin from the point of failure and work toward success.

> **Note:** You don't have to use a test-*driven* approach to get the benefit of unit testing. You can always write your tests after you've written the code. But I think you'll find that the more you use unit tests, the more you use the tests as a starting point. And you'll probably write better tests. The danger of writing tests for existing code is you

tailor the test to the code, rather than create tests to exercise all of the possible uses of the code.

The great thing about doing test-first development is you tend not to get hung up on the implementation aspects. You think about how you will want to use the code, and once you have that sorted out the implementation tends to come more quickly.

Of course over time you'll find other, better ways to implement that functionality. Rewriting code can be scary, but if you have a good suite of unit tests for that code you can rewrite with confidence, knowing that your code still passes its tests.

Extending ClarionTest

The first few versions of ClarionTest had a couple of bugs, notably that you couldn't rerun a test without first reloading the DLL.

The problem, you see, was that I didn't *have* ClarionTest while I was *writing* ClarionTest. But now that I do have ClarionTest I've refactored ClarionTest's own code using a test-driven approach. That's cleared up a bug or two and greatly improved the reusability of the code.

There's now a top-level `ClarionTest_TestRunner` class which you can use to load a DLL, discover test procedures, and execute test procedures. What you do with the results of those tests is up to you. I know down the road I'd like to add some batch capability so I can run tests on a set of DLLs and get back a report of all the test results.

The `ClarionTest` class library is beyond the scope of this article; I'll go into it in some detail in an upcoming article. Meanwhile, if you're curious have a look at the ClarionTestTest app in the source download.

Summary

It came as a surprise to me that it's been a whole year already since the first release of ClarionTest. I've been using it much more intensely in the past couple of months, and that's prompted me to make some changes including the new, compact `AssertThat` syntax and the removal of the need to prefix test procedures with `TEST_`.

I'm hearing from other Clarion devs who have started using ClarionTest, and I trust that number will continue to grow. If you have any suggestions for improvements, or you have some code you'd like to contribute, please let me know.

I've included a download link here, but you may also want to check the Google Code repository for the latest version (0.3 or better).

I also suggest you head over to ClarionLive and watch Webinar #87 on ClarionTest and Test Drive Development, which I recorded on Dec 10, 2010.

Download the ClarionTest 0.3 source

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).

Article comments

BACK TO TOP