Clarion Magazine

Clarion News

- » PrintWindow Build 103
- » Dan Pressnell's Better OOP Series
- » DevDawn Facelift
- » CapeSoft Profiler
- » RVCS Version Control For Clarion 6
- » Clarion Market
- » PDF Form Data Demo App
- » X-Lib For Clarion 6.2 Coming Soon
- » ClarionFAQ Back
- » Free UpdatePROTECT in SetupBuilder 5.2 Developer
- » PrintWindow 1.00 Beta Build 98
- » PowerOffice Web Site Jazzed Up
- » SetupBuilder 5.2 Build 1320
- » CDC Week #10 Winner
- » CPCS Support Unavailable 11/14/2005 - 11/19/2005
- » Ripley Code Commentor 1.19.1
- » TaskControl Released
- » CPCS 6.20 Build
- » SendTo XML Requires xFiles 1.03
- » ClarionJobs.com
- » CDC Week #9 Winners
- » Northern New Jersey CUG 11/17
- » Application Broker SE Available
- » IP Driver Demo Updated to Version 2
- » iQ-XML 1.11

- » UK User Group Meeting Nov 21, 2005
- » xXPframe 1.0
- » CDC Week #8 Winner
- » Clarion.NET hand coder's beta in sight?
- » cpTracker Professional 1/2 Price Sale
- » IP Driver Blog
- » In-Memory Driver Blog
- » IP Driver/Server and In-Memory Driver 2.0 Releases, New Online Store
- » FullRecord 1.10
- » xWordCOM 1.6
- » Dr.Explain Adds Clarion Control Support
- » Free xFunction Library 2.4
- » Clarion Developer's Challenge Home Page
- » Week #7 - Ten Lucky Winners!
- » PrintWindow 1.00 Beta Build 97
- » TipLink Tool Suite adds TipLink Navigator for CapeSoft's Ezhelp

[More news]

Podcast

[Track lists, more podcasts]

Latest Free Content

[More free articles]

Clarion Sites

- » SoftVelocity Company Site
- » SoftVelocity Community Site
- » SoftVelocity Online Store
- » CapeSoft

Clarion Blogs

- » Clarion Magazine
- » Gary James
- » DevDawn
- » Lesley Dean
- » SoftVelocity
- » JohnMo
- » Knobblegrud

Latest Subscriber Content

Exporting APPs and DCTs to XML, Part 2

XML is everywhere these days; everywhere, that is, but in your DCTs and APPs. But as Harley Jones shows, exporting your APP or DCT to XML can be a great benefit, for documentation, and for other purposes. Part 2 of 2.

Posted Friday, November 04, 2005

A Tree in a Page Loaded Browse: the Sequel, Part 2

Browse boxes formatted as trees are handy for all sorts of things. File loaded trees are easy; page loaded trees, however, can be a nightmare. In this series David Podger and Deon Canyon build on an earlier approach demonstrated by Ronald van Raaphorst. Part 2 of 3.

Posted Friday, November 04, 2005

PDF for October 2005

All Clarion Magazine articles for October 2005 in PDF format.

Posted Tuesday, November 08, 2005

A Tree in a Page Loaded Browse: The Sequel, Part 3

Browse boxes formatted as trees are handy for all sorts of things. File loaded trees are easy; page loaded trees, however, can be a nightmare. In this series David Podger and Deon Canyon build on an earlier approach demonstrated by Ronald van Raaphorst. This final installment looks at cutting and pasting.

Posted Friday, November 11, 2005

Memory Mapped Files in Clarion

Sometimes you need a secure way to communicate between applications, as Marty Honea discovered when he wanted a single login for multiple EXEs. After considering a variety of schemes, Marty found that Memory Mapped Files fit the bill perfectly.

Posted Friday, November 18, 2005

.NET Basics: What Is .NET, And Why Should I Care? Part 2

In Part 1 of this series Dave Harms discussed some basic .NET concepts such as the .NET framework, IL code, and namespaces. In this installment Dave surveys the classes that make up the .NET framework, and elaborates on the concept of namespaces.

Posted Thursday, November 24, 2005

A FileDropBox With Conditional Content

There are always situations where the contents of a FileDropBox, either a FileDropListBox or a FileDropCombo, is dependant on some criteria, usually the contents of an entry field or another FileDropBox. But a FileDropBox is always file loaded, and this loading is done when the form opens; after that the FileDropBox remains pretty static. Maarten Veenstra shows how to change the FileDropBox's content at runtime.

Posted Wednesday, November 30, 2005

Clarion Developer Bio: Sim Sherer

From 2001 to 2003, Sue Pichotta ran a series of Clarion developer bios for the IceTips News Network. We've been meaning to restart this series for some time; I trust Sim's bio will be the first of many.

Posted Wednesday, November 30, 2005

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

Printed Books & E-Books

E-Books

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

Clarion Magazine

Clarion News

Search the news archive

PrintWindow Build 103

New in this version of PrintWindow: Vertically expand listboxes to show complete content of the queue (valid only for file loaded listboxes). Controls under the listbox will move down, controls surrounding the listbox will resize.
Posted Monday, November 28, 2005

Dan Pressnell's Better OOP Series

The Better OOP series of articles by Dan Pressnell has been added to the articles section at www.clariondeveloper.net . The example files are also have been added to the download section. The article section is available to registered users only. Once you register you can also submit your own articles.
Posted Monday, November 28, 2005

DevDawn Facelift

Dev Dawn has taken on a new look and feel.
Posted Monday, November 28, 2005

CapeSoft Profiler

The CapeSoft Profiler will run you application, in the same way as a debugger does, and collect performance data for each line of your application source code. This allows you to easily locate the "slow" points in your application. To use the Profiler you simply compile your application with "full" debugging enabled. It is then ready for profiling. Demo

available.
Posted Monday, November 28, 2005

RVCS Version Control For Clarion 6

RVCS is a VCS system written for Clarion 6 using Clarion 6 and is intended for Clarion 6.x developers. RVCS interfaces on a command line level with Clarion 6 and is controlled by the Clarion 6 IDE VCS commands. RVCS allows a single or multiple programmers to make use of a single VCS repository for a project and implements module level READONLY locking of files already opened and in use by other developers. RVCS is in Beta and file formats may change or functionality may be added and other functionality depreciated as RVCS approaches Gold Release status. RVCS does not modify the APV/DCV files generated by the Clarion 6 IDE in any way. Demo available.
Posted Monday, November 28, 2005

Clarion Market

Clarion Market is an online resource for Clarion developers. You can ask questions, post answers, and browse news listings (the latter is Clarion Magazine's news feed).
Posted Monday, November 28, 2005

PDF Form Data Demo App

A demo app showing how to use the PDF-Tools SDK to bulk fill in PDF standard forms downloaded from the web is now available. This example uses the US Government Revenue Service Tax form 1099-R. The PDF page (or pages) from the original form are stored only once in the output file created - not multiple times - and therefore the file size is minimal. In the demo watermarks do somewhat compromise the final file size if you are testing an unlicensed version. The demo in question is called tx2pdf35.app and sample PDF forms and data are also provided (PDFOverlay1099.pdf and data.txt). Please note this is not an example of an Adobe 'Acroform' application.
Posted Monday, November 28, 2005

X-Lib For Clarion 6.2 Coming Soon

Rumors that the Dynamic File Driver has killed off X-Lib are premature. David Merenstein reports that SoftVelocity is currently testing X-Lib for Clarion 6.2.
Posted Monday, November 28, 2005

ClarionFAQ Back

A note from Jeff Slarve that ClarionFAQ.com is once again operational.
Posted Wednesday, November 16, 2005

Free UpdatePROTECT in SetupBuilder 5.2 Developer

SetupBuilder 5 Developer Edition includes a variety of features designed to help you manage access to your software, including serial number lists and date-based or version-based expiration. UpdatePROTECT is an integral part of the SetupBuilder 5.2 Developer Edition flagship now. You can quickly generate "Subscription Keys" to protect your updates. This technology is well suited for software subscription services in which the end-users subscribe, for instance on a monthly, quarterly, semi-annually, etc. basis. At the end of the subscription period, end-users will have the option of extending the subscription or simply continuing their current version. But end-users will no longer be able to utilize product updates released after the subscription expiration date/version. With UpdatePROTECT, the subscription renewal is easy and simple. You only have to send a new "Subscription Key" to your end-user. If you distribute software to others - within a department, company, or to the public - ensuring that your users can install the software smoothly should be a major concern. SetupBuilder 5 can help ease your mind. SoftVelocity, Inc. uses SetupBuilder to create service packs (patches) for Clarion 6.2. The size of the entire set of Clarion 6.2 "Gold" update files is more than 50 MB. The SetupBuilder byte-level patching mechanism reduces this to 3.6 MB (93% reduction) to keep Internet bandwidth to a minimum.
Posted Wednesday, November 16, 2005

PrintWindow 1.00 Beta Build 98

PrintWindow 1.00 Beta Build 98 has optional generation warning error to identify controls without an equate (such controls do not print).
Posted Wednesday, November 16, 2005

PowerOffice Web Site Jazzed Up

The PowerOffice web site has a new look, with nice graphics done by Sean Cameron.
Posted Wednesday, November 16, 2005

SetupBuilder 5.2 Build 1320

SetupBuilder 5.2 Build 1320 is now available. Click on "Check for updates" in the Help

menu to auto-update your current SB5 version. Changes include: Added "Start page" which, as the name implies, is presented when SB5 first loads, lets you you select a project to load from the list of previous projects, or start a new one; Added "Enable Installer Integrity Check" feature to help provide detection of tampered, hacked, and incomplete or virus infected installation files; Added "IIS Version" option to the "Get System Information" script function to retrieve the installed Internet Information Services (IIS) version; IDE adds full support for the Large Fonts mode (120 dpi); IDE enhanced "Quick Menu"; Added the "undocumented" command line options /SC and /DS to the documentation; If text resource is out-of-date, the IDE can automatically update the text resources; Installer "Retry" option added to overcome temporary file lock problems (caused by some virus-protection and anti-spy protection applications); Several bug fixes.
Posted Wednesday, November 16, 2005

CDC Week #10 Winner

This week's Clarion Challenge winner is Jim Henry. Jim wins OutlookBar from Gøran Thomassen of PowerOffice.
Posted Wednesday, November 16, 2005

CPCS Support Unavailable 11/14/2005 - 11/19/2005

Larry Teames will be out of the office 11/14/2005 thru 11/19/2005. All emails and support issues will be addressed as quickly as possible after the 19th.
Posted Wednesday, November 16, 2005

Ripley Code Commentor 1.19.1

Version 1.19.1 of the Ripley Code Commentor includes the following changes: XP Manifest has been turned on, for an XP look; Frame dragging has been turned off; Single line comments are now supported (this is the default after the update - if you don't want it you must turn if off); The "Check Clarion" procedure that ensures the block copy/delete settings are correct now checks the Automatic Block Delete setting as well as the Remove Block on Copy setting at startup.
Posted Wednesday, November 16, 2005

TaskControl Released

This new program from Huenuleufu Development monitors your running tasks and automatically kills CPU-hogging processes. There are two versions of the program,

TaskControl Lite and TaskControl Pro. TaskControl Lite shows the running tasks in a window monitoring the CPU % used. You can set a CPU % hog limit, the number of cycles and the refresh rate for each cycle that the condition should met before the task is automatically terminated. This allows you to monitor hung or badly terminated programs that remain in memory. In addition, you can manually kill a process just like in Window's task list. TaskControl Pro lets you define which processes to monitor by name. For each particular process, you can define a CPU % limit or a CPU time used limit, and a number of cycles that condition should met in average. You can also define an executable that should be launched after the program is terminated (allowing unattended restart of the terminated program). All operations and errors are logged to a special file, so you can later inspect what happened in your absence. With this version, it is impossible to terminate the wrong program by accident. Both products are available through Clarionshop.
Posted Wednesday, November 16, 2005

CPCS 6.20 Build

The new CPCS 6.20 build adds an option to allow hiding the "Print This Page" icon on the CPCS Preview (see the Advanced (cont'd) Tab). Existing v6.20 install codes will work for this build as well.
Posted Wednesday, November 16, 2005

SendTo XML Requires xFiles 1.03

A reminder from Geoff at CapeSoft that those wanting to utilize the new SendTo XML feature will require the new version of CapeSoft xFiles (v1.03). xFiles is not required with SendTo, but gives you the additional export to XML capability. SendTo FTP (out the box) will be available shortly as well. Full icon support should be in available in the next release (includes internal icons and all color-depth ranges).
Posted Wednesday, November 16, 2005

ClarionJobs.com

As the name suggests this iss a website dedicate to Clarion jobs. As a programmer you can register for free listing your skills and type of employment you are looking for. As an employer you can register for free and search for programmers based on your required skills, ideal if you are looking to fill full time positions or you need extra resource on a temporary basis. You can post projects on the site where registered programmers can submit quotes. ClarionJobs also offers adjudication service, escrow facilities, and project management. ClarionJobs is a division of Intelligent Silicon Ltd and is based in Bishops Stortford, Hertfordshire, in England.

Posted Wednesday, November 16, 2005

CDC Week #9 Winners

Michael Frazer and Nelson Andre Hofer de Carvalho are winners in week nine. Michael receives his choice of BoxSoft products (up to $299 value), and Andre wins his choice of Huenuleufu Development products (up to $149 value).
Posted Wednesday, November 16, 2005

Northern New Jersey CUG 11/17

The Northern New Jersey Clarion Users Group will be holding an open meeting this Thursday (11/17) at 6:30pm at the Holiday Inn in Saddle Brook, NJ. For details email ccordes at T R V A . com.
Posted Wednesday, November 16, 2005

Application Broker SE Available

This edition of the Application Broker features a brand new hi-performance multi-threaded socket engine, built-in SSL support, installs and runs as a Service and has remote administration.
Posted Thursday, November 10, 2005

IP Driver Demo Updated to Version 2

Clarion Web Hosting has updated the IP driver demo app so it now runs on version 2. There is also a new Feedback area where you can leave your comments. All the records from the original IP Driver have been copied across so if you previously registered your username and password are still valid. You can download the version 2 from http://www.clarionwebhosting.com/ipinvoice.exe. If you would like to do a side by side comparison with the IP Driver 1 demo using the original IP Driver you can download it from http://ipdriver.clarionweb.com/ipinvoice.exe. If you experience any problems feel free to email richard isv1 com.
Posted Tuesday, November 08, 2005

iQ-XML 1.11

iQ-XML 1.11 includes a few fixes and changes: All the Writer functions were not using the proper maximum field length limitation of 2048 bytes, but was using either 255 or

1024 (all increased to 2K except for CDATA); CDATA was not being written out properly for XML:AddElement (and can now handle 64K blocks); Change in code made Writer functions faster and more efficient. iQ-XML is a free XML Parser and Writer (with many neat features) for Clarion 5.5 and 6.x
Posted Tuesday, November 08, 2005

UK User Group Meeting Nov 21, 2005

If you're a third party provider and you want to include any flyers with the packs or have demo programs/PowerPoint presentations included on the user group CDs which every attendee gets please email Richard@isv1.com. Less than a third of the user group's members use the news groups to keep up to date with what's happening in Clarion so this is a great way to reach more people.
Posted Tuesday, November 08, 2005

xXPframe 1.0

xXPframe 1.0 is a class and code template that adds an XP-style frame menu to your application. xXPframe features include: Enable/disable Office XP menu style; Support for rising icon on select; Support for normal and gray icons for item; Set and change font and colors for menu; Merging menu; Support for window list (STD:WindowsList); Support single EXE, Multi-DLL (Local Mode, Standalone Mode), 32-bit. Compatible with Clarion 6.2 (build 9047-9048), Clarion 6.1 (build 9034). Compatibility with Clarion 5.5 is possible, but is not guaranteed. Price: $59.
Posted Tuesday, November 08, 2005

CDC Week #8 Winner

A big thank you to this weeks sponsor, Gøran Thomassen of PowerOffice, for donating PowerToolbar. This week's winner of the Clarion Developers Challenge on-line football office pool is Peter Kompier.
Posted Tuesday, November 08, 2005

Clarion.NET hand coder's beta in sight?

Bob Z has posted a message reiterating that the first release of Clarion.NET will be a hand-coder's edition, and will only be released to selected developers.
Posted Monday, November 07, 2005

cpTracker Professional 1/2 Price Sale

The cpTracker Professional 1/2 Price Sale is on through the end of 2005 just for Clarion developers. To purchase, please visit www.berthume.com/purchase.htm and select the Custom Solution option. Enter $49.00 USD. Regularly $99. For every two user licenses purchased, get the 3rd free.

Posted Tuesday, November 01, 2005

Clarion Magazine

Exporting APPs and DCTs to XML, Part 2

by Harley Jones

Published 2005-11-04

Okay, so far, I've touched on several intermediate and advanced Clarion Template techniques. And I've built a good, simple #GROUP library for generating XML from the templates. The DCT file is pretty much exhausted, so let's move on to the APP file. I broke it up into some logical portions, in the same manner as the DCT file:

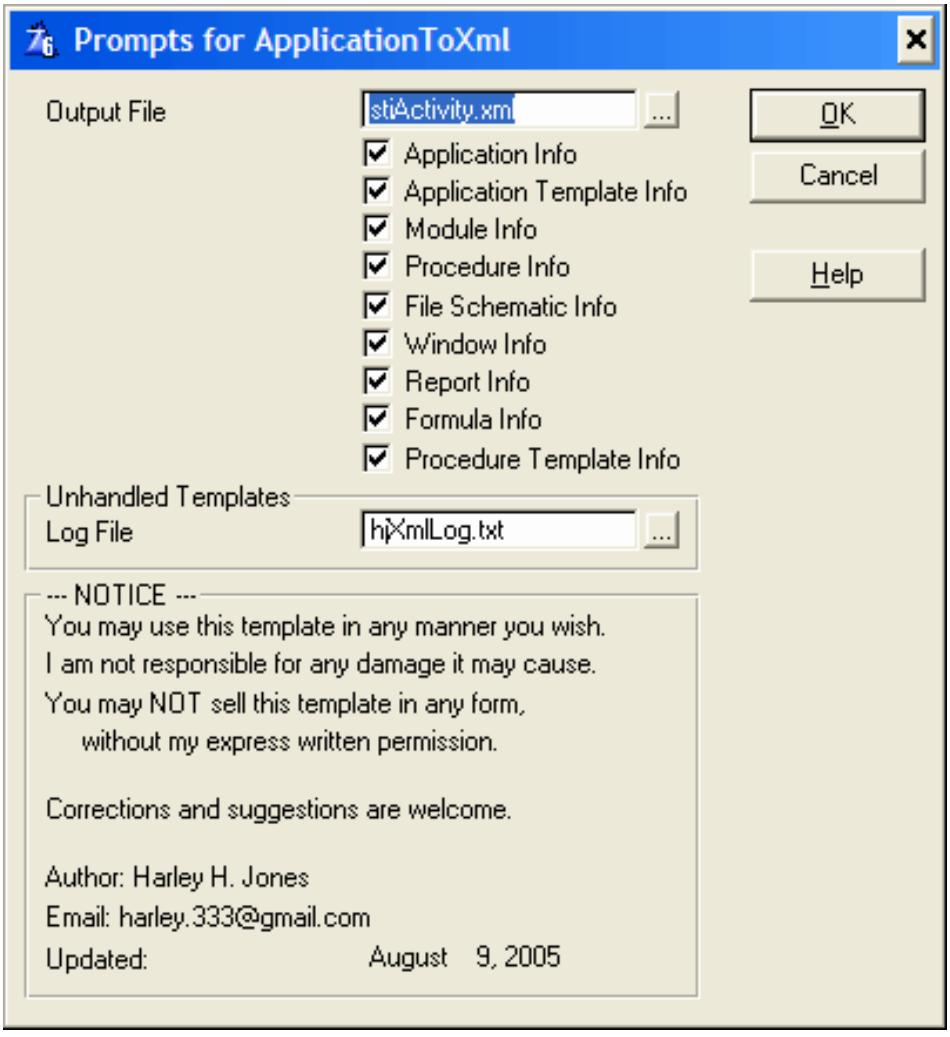


Figure 3. The application export template prompts

I won't go into detail about this template. It follows the same conventions that I used for the DCT file. But the APP file introduces a unique challenge that I was not able to overcome. Templates can create their own symbols using #PROMPTs. These symbols get saved within the APP file, and they are retrievable for use later. But they are only retrievable if you know how to find them. Unfortunately, the Clarion template symbols are not discoverable. I built a method for retrieving these symbols, but it requires that a developer write a #GROUP to handle each template. For instance, here's the code to handle Application-level #EXTENSION templates:

```
#IF(%pExportApplicationTemplateInfo)
  <Templates>
  #FOR(%ApplicationTemplate)
    <Template>
    #INSERT(%GetApplicationTemplateInfo)
    </Template>
  #ENDFOR
  </Templates>
#ENDIF
```

And here's the #GROUP(%GetApplicationTemplateInfo):

```
#GROUP(%GetApplicationTemplateInfo)
%(%WriteTag('Name', %GetStringValue(%ApplicationTemplate)))
<Instances>
#FOR(%ApplicationTemplateInstance)
  <Instance>
    %(%WriteTag('InstanceNumber', %ApplicationTemplateInstance))
    %(%WriteTag('Description',
        %GetStringValue(%ApplicationTemplateInstanceDescription)))
    %(%WriteTag('ParentInstance', %ApplicationTemplateParentInstance))
    %(%WriteTag('PrimaryInstance', %ApplicationTemplatePrimaryInstance))
    %(%HandleApplicationTemplate(%ApplicationTemplate))
  </Instance>
#ENDFOR
</Instances>
```

This is far as my code can go without specific information about external templates. You may have noticed at the very top of my template code, I include a TPW:

```
#INCLUDE('hjXmlHandler.TPW')
```

This TPW includes the two #GROUPs to handle external Application-level and Procedure-level #EXTENSION templates:

```
#GROUP(%HandleProcedureTemplate, %sTemplateName), PRESERVE, AUTO
  #ADD(%UsedTemplates, %sTemplateName)
```

```
    #SET(%UsedTemplateIsHandled, %True)
    #CONTEXT(%Procedure, %ActiveTemplateInstance)
      #CASE(%sTemplateName)
      #ELSE
        #SET(%UsedTemplateIsHandled, %False)
      #ENDCASE
    #ENDCONTEXT
  #!
  #GROUP(%HandleApplicationTemplate, %sTemplateName), PRESERVE, AUTO
    #ADD(%UsedTemplates, %sTemplateName)
    #SET(%UsedTemplateIsHandled, %True)
    #CONTEXT(%Application, %ApplicationTemplateInstance)
      #CASE(%sTemplateName)
      #ELSE
        #SET(%UsedTemplateIsHandled, %False)
      #ENDCASE
    #ENDCONTEXT
```

As you can see, these #GROUPs do practically the same thing. First, they add the current external template name to a collection of all the template names used in the APP (%UsedTemplates). The collection has a dependent symbol, %UsedTemplateIsHandled. This symbol indicates whether or not someone has supplied code to retrieve the symbols from the current external template. The #CASE statement expects to find the template name and run the proper code for the external template. The #CONTEXT line is the fancy part. Here's what the help has to say about #CONTEXT:

> The key to understanding the use of the #CONTEXT structure is "as if the source code for the named section were being generated." This statement means that the statements are evaluated as if #GENERATE were executing. For example, a #EMBED statement within a #CONTEXT structure does not name a new embed point but instead, generates the contents of the named embed point.

This means that everything about the current template is now free for the taking. All #PROMPTs, #DECLAREd and #EQUATEd symbols, and even #EMBEDded code are now in scope and can be swiped with ease. All you need to know is the name of each symbol!

At the end of my template, I generate a log file containing the names of all the external templates that were not handled. So, it's easy to know where the holes are. Filling those holes are as easy as creating a new #GROUP, use the #GROUPs I've talked about in this article to generate XML, and #CALL your new #GROUP from the #CASE statement.

I've touched on several intermediate and advanced template techniques in this article. I'm sure I didn't include as much detail as some of you would like. Please drop a comment or a question, and I'll be sure to answer it. In the next article, I'll demonstrate how to use XSLT to transform the generated XML files into something useful.

To summarize, I've covered:

- #CONTEXT
- #EXTENSION vs #UTILITY
- #DECLARE vs #EQUATE
- PROP syntax
- Multi-value symbols
- Dependent symbols
- #PREPARE section
- #CREATE and #CLOSE
- #GROUPs
- #CREATE vs #OPEN
- #MESSAGE

Reference variables in templates

Since I've finished the templates, I've used them several times for all sorts of things. Obviously, documentation is an easy fit. But one of the best uses was much more complicated. The company I work for writes administrative software for schools. There are a ton of state mandated forms. In the past, we've laid out the screens and the reports separately. Plus, the table structures needed to be laid out. When everything was in Clarion, it was a pain to keep the three areas in sync. But there was compile-time checking, so at least we could take comfort in that.

When customers started asking for web solutions, there was a fourth layout that we decided to do by hand (this time without the compile-time checking). Things always go bad when you're coding the same info in several places, and I saw it as a huge inefficiency. When the requirement came down that several of these forms needed to be migrated to .Net screens and our .Net reports (ActiveReports), I convinced the powers that be that there was a better way. We now use the Clarion report layout to generate seven different kinds of information: Clarion screens, .Net screens (these are actually VB.Net code), .Net reports, ASCX, CSS, HTML, and table structures. And each of the layouts is absolutely pixel-perfect! Now, when a customer asks "Do you think we could see a demo of some web-enabled forms?", we can generate their forms on the web in seconds.

Of course there were interesting problems: going from THOUS to twips, THOUS to DLUs, THOUS to pixels; our .Net reports only support a single detail band; and we needed to generalize our screen logic. But along with all of that refactoring, we found tons of places to optimize our development process. For example, in Clarion, screen behaviors are coded singly. (A screen behavior is my term for something like, "When you check this box, this textbox is required." It's not a business or validation rule, it's just a usability hint.) But now that all the screen layouts could potentially run within a generic engine, why couldn't screen behaviors? We invented an XML schema to store screen behaviors, and we use the Windows script engine to run Javascript that rips through the XML. Why Javascript, you ask? Because it also works on the web! Our screens and 90% of the code necessary to make them usable is now platform independent. (By the way, the script engine is still not implemented in Clarion. The powers that be haven't deemed it useful enough yet. So we still write a bunch of duplicate code. Sooner or later, I'll implement it and that'll probably be another article.)

Suffice it to say, I see XSLT as a very simple code generator, not nearly as complex or complete as

Clarion, but dirt simple to write. And I'll have more to say on XSLT in the next article.

Download the source

Harley Jones lives in Mobile, AL. He's been programming in one way or another since he was ten, and seems to have a knack for it. After graduating college with a degree in English Literature, he was hired as a programmer, and was introduced to Clarion. Currently, he's buried in .Net projects where he's constantly looking for methods to streamline production with code generators.

Reader Comments

Add a comment

- » Great work Harley... I can't wait for the next article....

Clarion Magazine

A Tree in a Page Loaded Browse: the Sequel, Part 2

by David Podger and Deon Canyon

Published 2005-11-04

Last week we introduced our enhancement to Ronald van Raaphorst's "A Tree in a Page Loaded Browse." Now it's time to see how the code works.

To begin an explanation of the code, we start with a record highlighted in the Plays browse and an *empty* Tree browse. The `NoTree` variable is `TRUE` and the New Script button is visible, as explained above. The user presses the New Script button to begin adding records entries to the Tree. The embed point is shown in Figure 15.

Figure 15. Embed at Control Events, ?ButtonScript

The inserted code is:

```
IF RECORDS(CutQu)
  DO UndoTags                 ! cancel pending cut or copy
END
IF NoTree
  GLO:Play  = PLA:PlayAuto    ! if no Tree records, GLO:Play
  GLO:SeqNo = ''              ! is primed from Plays record
ELSE
  GLO:Play  = TRE:PlayAuto    ! otherwise, priming is from
  GLO:SeqNo = TRE:SeqNo       ! the highlighted Tree record
END
GLO:Type  = 'Script'
POST(Event:Accepted,?Insert:3) ! the hidden, real Insert button
```

The `CutQu` holds cut or copied tree rows which are waiting to be pasted. The cut and paste routines are described later.

The rest of the code places values in three variables: `GLO:Play`, `GLO:SeqNo` and `GLO:Type`. These variables could be parameters passed to the `UpdateTREE` procedure. We have preferred to use Globals so that the standard link to an update form (which, in Clarion version 5, does not allow parameters) can be used. In the van Raaphorst articles there is an explanation of how to parameterize this update. Either way it is done, the update procedure must obtain the identity of a highlighted record in the Tree or know the Tree is empty. This information allows it to build an appropriate `SeqNo` for the inserted record. Secondly, `UpdateTREE` must know which record type is to be inserted, so it can customize the appearance of the update form. It does this by exposing different wizard-style tab sheets on the form.

We now turn to a description of what happens within the `UpdateTREE` procedure.

On insert, the three globals are used as follows: Two of the fields (`TRE:PlayAuto` and `TRE:TreeType`) are primed as shown in Figure 16.

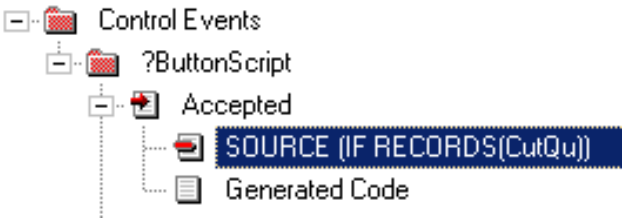


Figure 16. Field Priming on Insert

`TRE:SeqNo` is constructed using GLO:SeqNo as a starting point. The code to do this construction work is embedded as shown in Figure 17.

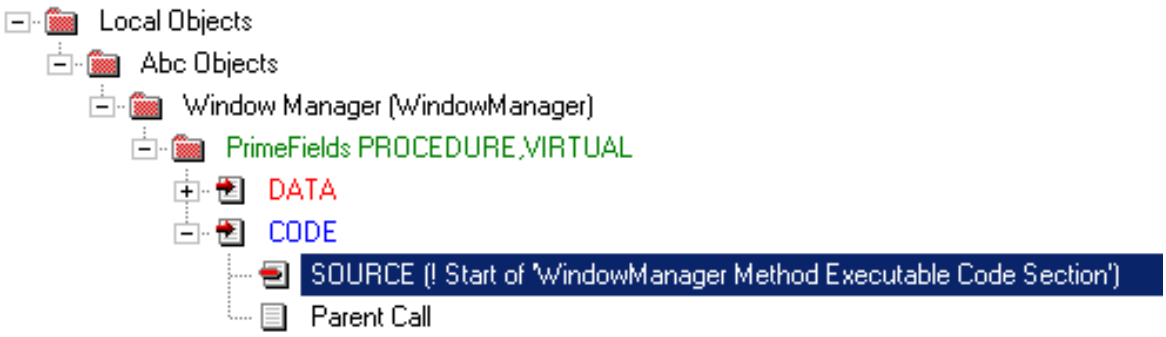


Figure 17. Constructing TRE:SeqNo

The embedded code is as follows:

```
CurSeqNo  = GLO:SeqNo
SaveID    = Access:TREE.SaveFile()
IF GLO:Type = 'Script'
  TRE:PlayAuto = GLO:Play
  TRE:SeqNo = '9999'
  SET(TRE:Key_SeqNo,TRE:Key_SeqNo)
  IF Access:TREE.Previous() = LEVEL:Benign
```

```
      NewSeqNo = TRE:SeqNo[1 : 4] + 1
    END
    Access:TREE.RestoreFile(SaveID)
    TRE:SeqNo = FORMAT(NewSeqNo,@N04)
  ELSE
    TRE:PlayAuto = GLO:Play
    TRE:SeqNo = GLO:SeqNo & '.9999'
    SET(TRE:Key_SeqNo,TRE:Key_SeqNo)
    IF Access:TREE.Previous() = LEVEL:Benign AND |
        SUB(TRE:SeqNo,1,LEN(GLO:SeqNo)) = CurSeqNo
      IF LEN(TRE:SeqNo) <> LEN(GLO:SeqNo)
        NewSeqNo = SUB(TRE:SeqNo,LEN(GLO:SeqNo)+2,4) + 1
      END
    END
    Access:TREE.RestoreFile(SaveID)
    TRE:SeqNo = CurSeqNo & CHOOSE(CurSeqNo = '','','.') & |
        FORMAT(NewSeqNo,@N04)
  END
```

Please refer to Ronald van Raaphorst's articles for a detailed explanation of how this code works. What is shown above is a close adaptation of his code to the needs of our example. It works for all record types. That completes the extra work needed *on the update form itself* to make it work as the update of a tree file. As can be seen, there was not much to do.

Returning to the browse, there is nothing out of the ordinary to do either before or after a change record. For deletes, however, the children of a deleted record must be dealt with. In our example we have chosen to cascade the delete and remove any attached child records (they are not child records in the strict sense, since they are all in the one file – but they are logically attached). We have chosen to delete the attached records before deleting the highlighted record, as shown in Figure 18.

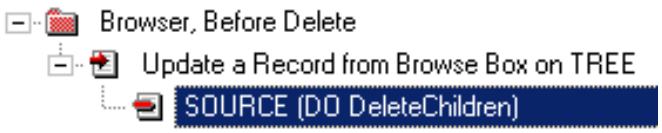


Figure 18. Embed to delete children

The routine's code is as follows:

```
  DeleteChildren        ROUTINE
    GET(TREE,TRE:Key_SeqNo)     ! no error check needed
    LenSeq   = LEN(TRE:SeqNo)
    BasePlay = TRE:PlayAuto     ! save to re-get parent
    BaseSeq  = TRE:SeqNo        ! at the end of routine
    STREAM(TREE)
    SET(TRE:Key_SeqNo,TRE:Key_SeqNo)
    NEXT(TREE)                  ! reads parent again
    NEXT(TREE)                  ! reads first child
```

```
LOOP WHILE (NOT ERRORCODE() AND LEN(TRE:SeqNo) > LenSeq)
  DELETE(TREE)
  NEXT(TREE)
END
FLUSH(TREE)
TRE:PlayAuto = BasePlay     ! re-get parent for deletion
TRE:SeqNo    = BaseSeq
GET(TREE,TRE:Key_SeqNo)
```

The attached children of a record all have a `SeqNo` length that is *longer* than their parent. This single fact is the key to the above code. `LenSeq` is primed with the length of the parent. The code then positions on what should be the first attached child and deletes one record at a time until `LEN(TRE:SeqNo)` is less than or equal to `LenSeq`, or end of file is reached. This neatly removes all the children. It also leaves gaps in `TRE:SeqNo`, but this does not matter.

The code applies single-user logic and minimizes unnecessary error checking. It does not check for errors on reading records that prior code has determined are present. After the deletes, the browse needs to be correctly re-displayed, as shown in Fifure 19.

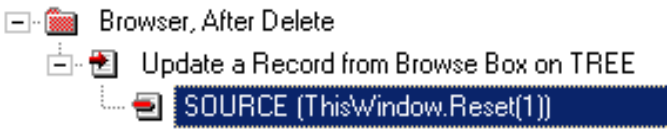


Figure 19. Correctly re-displaying the browse

The full code is:

```
ThisWindow.Reset(1)
SELECT(?List:2)
```

Moving records up and down

The bottom left-hand corner of the browse window shows the two buttons displayed in Figure 20.

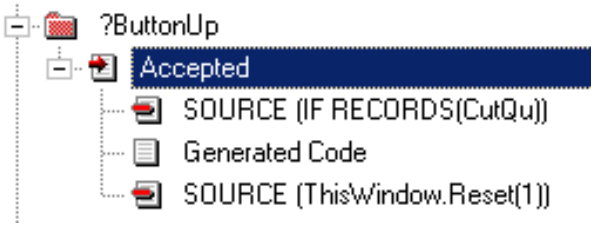


Figure 20. The Up and Down buttons

These are the buttons that move parts of the tree up or down. The embeds points are shown in Figure 21.

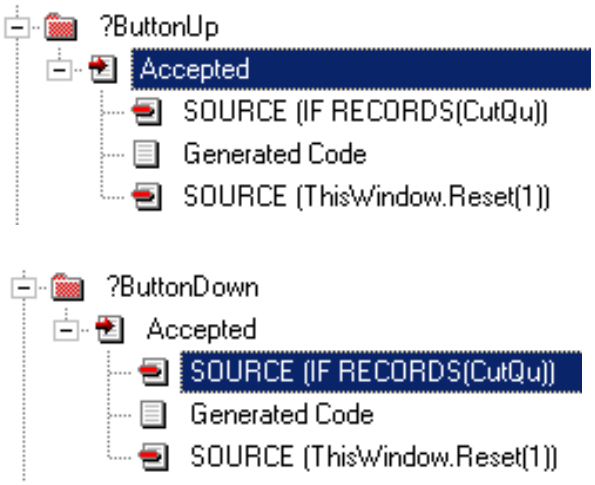


Figure 21. Embed points for Up and Down buttons

The full code for both embeds is:

```
IF RECORDS(CutQu)        ! cancel any pending cut or copy
  DO UndoTags
.
Direction = 'Up'               ! or  Direction = 'Down'
GET(TREE,TRE:Key_SeqNo)
DO Nudge
ThisWindow.Reset(1)
```

As can be seen from the above, the Nudge routine does both the up and down moves. It converts any move Down into a move Up of the record(s) below. The example used earlier is shown in Figure 22.

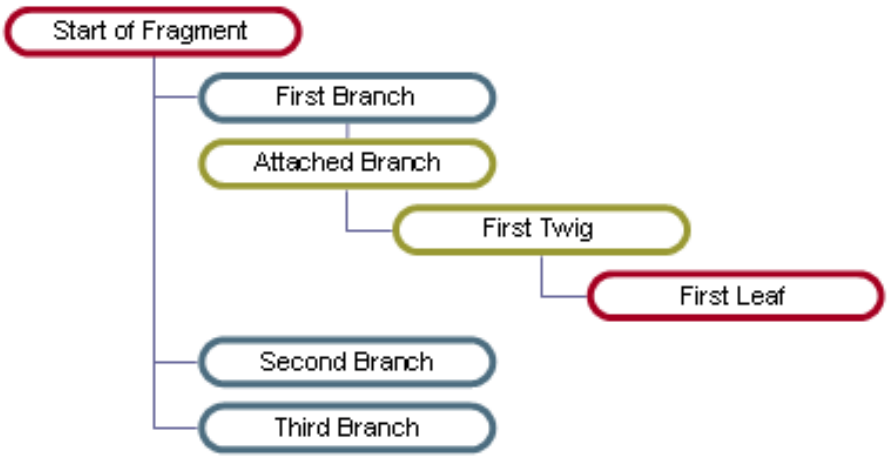


Figure 22. Tree Fragment

It does not matter whether the user highlights **First Branch** and moves it Down or highlights **Second Branch** and moves it Up. The exact same logic can be used. In effect, a swap occurs and **First Branch** and **Second Branch** exchange places.

Using the above example, the Nudge code divides into five logical parts:

1. If needed, convert move Down into move Up
2. Save **First Branch's** children (if any) to TreeQu, and delete from file
3. Swap the 'parents', **First Branch** and **Second Branch**
4. Change the SeqNo of **Second Branch's** children (if any) to move them up
5. Write **First Branch's** children from TreeQu and attach to parent

The code for the Nudge routine is as follows:

```
Nudge           ROUTINE
  STREAM(Tree)
  FREE(TreeQu)                ! empty a work queue
  LenSeq  = LEN(TRE:SeqNo)
  ! If a move Down, search down the file to get the Base record
  ! for a Nudge Up and do that instead

  IF Direction = 'Down'       ! if a Down, do it as an Up
    SET(TRE:Key_SeqNo,TRE:Key_SeqNo)
    NEXT(TREE)
    IF ERRORCODE()            ! already as far down as possible
      EXIT
    END
    NEXT(TREE)                ! start search for record to move up
    LOOP WHILE (NOT ERRORCODE() AND LEN(TRE:SeqNo) > LenSeq)
      NEXT(TREE)              ! continue search to end of attached
    END
    IF ERRORCODE()            ! fell off end, cannot move down
      EXIT
    END
  END
  LenSeq  = LEN(TRE:SeqNo)
  BasePlay = TRE:PlayAuto
  BaseSeq = TRE:SeqNo         ! hold base key
  SET(TRE:Key_SeqNo,TRE:Key_SeqNo)
  PREVIOUS(TREE)
  IF ERRORCODE()              ! nothing to move
    EXIT
  END
  PREVIOUS(TREE)
  ! While child above search higher, copy children, delete in
  ! file and find parent above. Change SeqNo in move to TreeQu

  LOOP WHILE (NOT ERRORCODE() AND LEN(TRE:SeqNo) > LenSeq)
    TreeQu = TRE:Record
    TQU:SeqNo[(LenSeq-3) : LenSeq] = BaseSeq[(LenSeq-3) : LenSeq]
    ADD(TreeQu)
    DELETE(TREE)  ! copy as BaseSeq to queue and delete from file
    PREVIOUS(TREE)
```

```
  END
  IF LEN(TRE:SeqNo) < LenSeq             ! exit: no swap possible
    EXIT
  END
  NewSeq = TRE:SeqNo           ! hold new key, ie key of parent row above
  DO SwapParentRows            ! do physical swap of the parent records
  TRE:PlayAuto = BasePlay      ! get back to base record
  TRE:SeqNo    = BaseSeq
  SET(TRE:Key_SeqNo,TRE:Key_SeqNo)
  NEXT(TREE)
  IF ERRORCODE()
    EXIT
  END
  NEXT(TREE)
  ! while child below Base, search until no children, swap to New
  LOOP WHILE (NOT ERRORCODE() AND LEN(TRE:SeqNo) > LenSeq)
    TRE:SeqNo[(LenSeq-3) : LenSeq] = NewSeq[(LenSeq-3) : LenSeq]
    PUT(TREE)
    NEXT(TREE)
  END
  ! write children to file from TreeQu
  LOOP T# = 1 TO RECORDS(TreeQu)
    GET(TreeQu, T#)
    TRE:Record = TreeQu
    ADD(TREE)
  END
  TRE:PlayAuto = BasePlay      ! highlight correct record in browse
  IF Direction = 'Down'
    TRE:SeqNo = BaseSeq
  ELSE
    TRE:SeqNo = NewSeq
  END
  GET(TREE,TRE:Key_SeqNo)
  FLUSH(Tree)
```

The above code executes another routine, `SwapParentRows`. It is self-explanatory:

```
SwapParentRows         ROUTINE
  NewSeq = TRE:SeqNo           ! save it in NewSeq
  TRE:PlayAuto = BasePlay
  TRE:SeqNo    = BaseSeq
  GET(TREE,TRE:Key_SeqNo)      ! get the base record again and
                               ! save it in a temporary location
  SaveSeq      = TRE:SeqNo[1 : (LenSeq - 4)] & '9999'
  TRE:PlayAuto = BasePlay
  TRE:SeqNo    = SaveSeq
  PUT(TREE)                    ! force elsewhere
  TRE:PlayAuto = BasePlay      ! get the Seq above
  TRE:SeqNo    = NewSeq
```

```
GET(TREE,TRE:Key_SeqNo)
TRE:SeqNo = BaseSeq          ! replace with base Seq
PUT(TREE)
TRE:PlayAuto = BasePlay
TRE:SeqNo    = SaveSeq
GET(TREE,TRE:Key_SeqNo)      ! get base record from 'elsewhere'
TRE:PlayAuto = BasePlay
TRE:SeqNo    = NewSeq
PUT(TREE)                    ! replace it with the NewSeq record
```

Next week we'll look at cutting and pasting.

Download the source

David Podger has been an independent Clarion developer in Australia for a decade. He presently lives in Sydney, where he sells a specialised accounting application for remote communities.

Dr Deon Canyon is a Senior Lecturer (translation: Associate Professor) in public health and medical entomology at James Cook University in North Queensland, Australia. His current research interests include: Disease control simulation models for use in distance education; *Lymphatic filariasis* (a disease caused by thread-like parasitic worms) and the *Aedes polynesiensis* mosquito in transmission and control of the Pacific Head Lice (*Pediculus capitis*); Health in remote settlements and the use of touch screen kiosks to improve it; Aedes aegypti mosquito behavior, physiology and ecology . Deon is married, and has three children. His hobby of choice is kite surfing.

Reader Comments

Add a comment

Clarion Magazine

A Tree in a Page Loaded Browse: The Sequel, Part 3

by David Podger and Deon Canyon

Published 2005-11-11

In part 1 of this series we introduced our enhancement to the tree handling approach described by by Ronald van Raaphorst in "A Tree in a Page Loaded Browse", and in part 2 we explained most of the source code. The one remaining area is cutting and pasting.

The process of cutting and pasting within a tree needs to be defined first. Obviously, a user allowed to free-form cut and paste could easily ruin a tree. We have found that, if the rows to be cut or copied are simply defined as *the attached children* of the highlighted row then, in practice, this gives adequate cut and paste functionality. Similarly, the paste target is best defined as the row where the cut or copied children *will be re-attached*. The parents do not move, just the children.

Now, it will not do if the user cuts or copies children attached to a Question and then pastes them under a record of a different type. This would destroy the tree's logic. The rule must be that the record *pasted to* must be of the same type as the original cut or paste. The paste target may be at a different *level* and it may or may not *have children*, these two conditions do not matter.

If there are existing children at the paste target, then the new ones are located *below* them, just as in an insert. The Up and Down buttons are there to nudge them to an exact location after they have been pasted.

The buttons are standard:

Figure 22. The standard cut/past/copy buttons

?ButtonCut and ?ButtonCopy have nearly identical embeds, as shown in Figure 23.

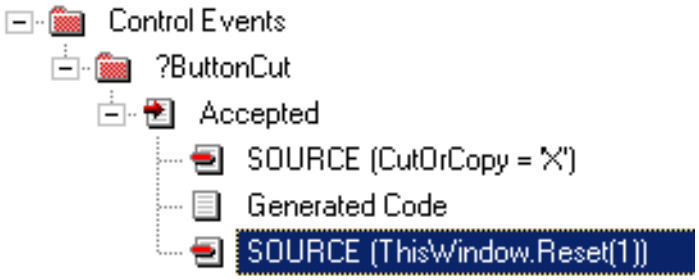


Figure 23. Embed points for Cut and Copy

The full code at the embed point before Generated Code is:

```
CutOrCopy = 'X'
DO TagRows
```

The code embedded after Generated Code is:

```
ThisWindow.Reset(1)
SELECT(?List:2)
```

The embedded code for copy is identical but for the one line: `CutOrCopy = 'C'`

The `TagRows` procedure begins by cleaning up the `CutQu` and any left over cut tags in the `Play`. Then, using the technique seen above in Nudge, it finds the children attached to the highlighted record, copies them to `CutQu` and remembers their original location and level.

```
TagRows              ROUTINE
  GET(TREE,TRE:Key_SeqNo)
  LenSeq  = LEN(TRE:SeqNo)
  BasePlay = TRE:PlayAuto
  BaseSeq  = TRE:SeqNo
  IF RECORDS(CutQu)                    ! remove any existing tags
    FREE(CutQu)                        ! remove cut queue
    STREAM(TREE)
    CLEAR(TRE:Record)
    TRE:PlayAuto = BasePlay
    SET(TRE:Key_Auto,TRE:Key_Auto)
    LOOP
      NEXT(TREE)
```

```
      IF ERRORCODE() THEN BREAK.
      IF TRE:PlayAuto <> PLA:PlayAuto
        BREAK
      END
      IF TRE:TreeTag <> ''
        TRE:TreeTag = ''
        PUT(TREE)
      END
    END
    FLUSH(TREE)
  END
  STREAM(TREE)
  TRE:PlayAuto = BasePlay
  TRE:SeqNo    = BaseSeq
  SET(TRE:Key_SeqNo,TRE:Key_SeqNo)
  NEXT(TREE)                                ! reads the parent
  IF ERRORCODE()
    FLUSH(TREE)
    EXIT
  END
  CutType = TRE:TreeType
  NEXT(TREE)                                ! reads first child
  LOOP WHILE (NOT ERRORCODE() AND LEN(TRE:SeqNo) > LenSeq)
    TRE:TreeTag = CutOrCopy
    PUT(TREE)
    CutQu = TRE:Record
    CUT:TreeTag = ''                        ! remove tag
    ADD(CutQu)                              ! save in cut buffer
    NEXT(TREE)
  END
  FLUSH(TREE)
  IF NOT RECORDS(CutQu)
    MESSAGE('Nothing to Cut or Copy','NEED CHILD ENTRIES')
    EXIT
  END
  CutPlay  = BasePlay                       ! save location of cut
  CutSeq   = BaseSeq
  CutLevel = INT(LEN(BaseSeq) / 5) + 1    ! calculates level
  TRE:PlayAuto = BasePlay    ! highlight correct record in browse
  TRE:SeqNo = BaseSeq
  GET(TREE,TRE:Key_SeqNo)
```

The stage is set for the paste. Pasting is done as follows. At the Control Events embed, under `?ButtonPaste`, insert the following, as shown in Figure 24.

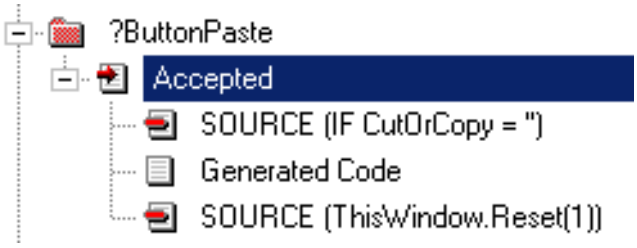


Figure 24. Embed point for pasting

The code that does pasting takes these steps:

1. Saves the location of the record highlighted when `?ButtonPaste` is pressed
2. Determines whether the paste record already has attached children and calculates the needed level increment to apply to all pasted children.
3. Saves last used `TRE:TreeAuto` so that additional `TreeAuto` values can have a correct starting value.
4. Checks that the paste record type matches the cut record, exits if error.
5. Warns the user if they are pasting back into the cut location, but allows this to happen.
6. Does the actual paste, using the `PasteTagged` routine.
7. If a cut and not a copy, deletes the records at the cut point.
8. If a copy, clears the copy tags from the copied records.

The full code for the embed *after* Generated Code is:

```
IF CutOrCopy = ''
  MESSAGE('There is nothing to Paste','EMPTY CUT BUFFER')
ELSE
  GET(TREE,TRE:Key_SeqNo)
  BasePlay = TRE:PlayAuto              ! save paste point
  BaseSeq  = TRE:SeqNo
  PasteLevel = INT(LEN(TRE:SeqNo)/5)+1 ! calculates level

  STREAM(TREE)
  SET(TRE:Key_SeqNo,TRE:Key_SeqNo)
  NEXT(TREE)                         ! read paste point again
  NEXT(TREE)                         ! read one past
  BumpPasteChild = 0
  LOOP WHILE(NOT ERRORCODE() AND LEN(TRE:SeqNo) > LenSeq)
    IF LEN(TRE:SeqNo) = LenSeq+5 ! child one level down
      BumpPasteChild += 1
    END
    NEXT(TREE)
  END
  FLUSH(TREE)
```

```
! BumpPasteChild: added to the Paste Level
! component of SeqNo for all pasted entries

SaveTreeAuto = 0
STREAM(TREE)
CLEAR(TRE:Record)
TRE:PlayAuto = BasePlay
SET(TRE:Key_Auto,TRE:Key_Auto)
LOOP
  NEXT(TREE)
  IF ERRORCODE() THEN BREAK.
  IF TRE:PlayAuto <> BasePlay
    BREAK
  END
  IF TRE:TreeAuto > SaveTreeAuto
    SaveTreeAuto = TRE:TreeAuto ! SAVE last used
  END
END
FLUSH(TREE)

TRE:PlayAuto = BasePlay         ! restore paste point
TRE:SeqNo = BaseSeq
GET(TREE,TRE:Key_SeqNo)         ! reget paste point
NewSeq = TRE:SeqNo[1 : (PasteLevel*5-1)]

IF TRE:TreeType <> CutType
  MESSAGE('Type mismatch - no PASTE')
ELSIF TRE:SeqNo = CutSeq
  CASE MESSAGE('paste to same location?',   |
    'PASTE IN SAME PLACE?',      |
    ICON:Question,Button:Yes+Button:No)
  OF Button:Yes
    DO PasteTagged
    IF CutOrCopy = 'X'
      DO DeleteTagged
    ELSE
      DO UndoTags
    END
  END
ELSE
  DO PasteTagged
  IF CutOrCopy = 'X'
    DO DeleteTagged
  ELSE
    DO UndoTags
```

```
    END
  END
END
```

The code below executes the `PasteTagged` routine. Its code is self-explanatory:

```
PasteTagged       ROUTINE
  T# = SaveTreeAuto
  LOOP R# = 1 TO RECORDS(CutQu  ! walk cut queue, changing SeqNo
    GET(CutQu,R#)               ! and PlayAuto
    ChildSeq = NewSeq  ! build new SeqNo, copy the paste Seq root
    L# = CutLevel * 5 + 1       ! calc the beginning of the first
    S# = CUT:SeqNo[L# : (L#+3)] ! child in the Cut
    C# = S# + BumpPasteChild    ! calculate the bump child level
    ChildSeq = ChildSeq & '.' & FORMAT(C#,@n04)    ! append it
    ChildSeq = ChildSeq & CUT:SeqNo[(L# + 4) : 255]! append  rest
    CUT:SeqNo = ChildSeq                           ! of cut SeqNo
    TRE:Record = CutQu  ! add the modified cut record to the Tree
    T# += 1             ! but create a new TreeAuto value first
    TRE:TreeAuto = T#
    ADD(TREE)
    IF ERRORCODE()
      MESSAGE(ERROR()).
  END
```

Lastly, the above code executes the `DeleteTagged` and the `UndoTags` routines:

```
DeleteTagged        ROUTINE
  STREAM(TREE)
  CLEAR(TRE:Record)
  TRE:PlayAuto = BasePlay
  SET(TRE:Key_Tag,TRE:Key_Tag)
  LOOP
    NEXT(TREE)
    IF ERRORCODE() THEN BREAK.
    IF TRE:PlayAuto <> BasePlay
      BREAK
    .
    IF TRE:TreeTag = 'X'
      DELETE(TREE)
    .
  END
  FLUSH(TREE)
```

```
UndoTags          ROUTINE
  GET(TREE,TRE:Key_SeqNo)
  BasePlay = TRE:PlayAuto
  SaveSeq  = TRE:SeqNo
  STREAM(TREE)
  CLEAR(TRE:Record)
  TRE:PlayAuto = BasePlay
  SET(TRE:Key_Auto,TRE:Key_Auto)
  LOOP
    NEXT(TREE)
    IF ERRORCODE() THEN BREAK.
    IF TRE:PlayAuto <> PLA:PlayAuto
      BREAK
    END
    IF TRE:TreeTag <> ''           ! if tagged, then
      TRE:TreeTag = ''             ! remove tag
      PUT(TREE)
    END
  END
  FLUSH(TREE)
  FREE(CutQu)
  TRE:PlayAuto = BasePlay   ! highlight correct record in browse
  TRE:SeqNo = SaveSeq
  GET(TREE,TRE:Key_SeqNo)
```

Conclusion

By completing the above code, we have succeeded in implementing Ronald van Raaphorst's solution to the point where it is reasonably serviceable. Drag and drop could be added, by borrowing from the cut and paste logic. The drag and the drop targets are both single records, so it would be simple to implement. The solution is single-user, since it is aimed at the designer of a dialogue. Hardening it for a multi-user implementation would be possible, but this is not a task we would wish to undertake.

Download the source

David Podger has been an independent Clarion developer in Australia for a decade. He presently lives in Sydney, where he sells a specialised accounting application for remote communities.

Dr Deon Canyon is a Senior Lecturer (translation: Associate Professor) in public health and

medical entomology at James Cook University in North Queensland, Australia. His current research interests include: Disease control simulation models for use in distance education; *Lymphatic filariasis* (a disease caused by thread-like parasitic worms) and the *Aedes polynesiensis* mosquito in transmission and control of the Pacific Head Lice (*Pediculus capitis*); Health in remote settlements and the use of touch screen kiosks to improve it; Aedes aegypti mosquito behavior, physiology and ecology . Deon is married, and has three children. His hobby of choice is kite surfing.

Reader Comments

Add a comment

Clarion Magazine

Memory Mapped Files in Clarion

by Marty Honea

Published 2005-11-18

Have you ever stood back and looked at one of your projects, thinking "that's nice but…"? Recently the "But…" I was dealing with had to do with multiple applications running on a machine, each oblivious to the existence of the others. In my programs, security is a major consideration, as I'm sure it is with many of yours. Access to the program is controlled by a login that dictates what kind of access each user has to features of the program. In my case, it is common for a client to have three or more of my applications running on their machine. Each program calls a login procedure, and I felt it was an inconvenience for my clients to have to log in three times to use my software. So, I set my mind to finding a way to use a single login, and as is almost always the case, simplifying the use of my software turned out to be more than I had bargained for. It turned out that Memory Mapping would be exactly what I was looking for.

Weighing my options

At first, I thought that a single login would be a simple thing to do. The first program would ask for a login, and then create a text file holding the login information; subsequent programs would use those credentials. I quickly decided there were too many problems with this approach. First and foremost was the worry of login information being stored in a format where it could easily be intercepted.

I moved on to considering an encrypted TPS file. This approach would afford a little more security than the text file, but (I really need to get that word out of my vocabulary) I would still need to delete the file when all the programs had terminated. How would I know when the user closed the last of my programs? This was a major hurdle.

After asking around to see what others were doing in this situation, I got pretty much the same response from everyone: "That would be a cool feature, but we don't do anything like that currently." I was on my own. Several ideas for handling the problem were tossed around on IRC chat. Someone suggested creating a critical section so any of my programs could determine if it was the last one running, and then delete the TPS file. This was a pretty good idea, but what happens if a program closes abnormally? "Closes abnormally," that's a fancy way of saying, what if it GPFs? I'm sure your programs don't close abnormally, but from time to time my programs do. There is no way to know if a Critical Section has been abandoned. I didn't want security to be compromised if my program decided to go to lunch early. A Mutex does however let you know if it has been abandoned. I needed to do more research.

While researching mutexes and critical sections, I came across a neat little thing called a Memory Mapped File. According to my early research Memory Mapped Files were exactly what I was looking for. A Memory Mapped File is nothing more than a section of memory you set aside that you access via a pointer to that section of memory. You can read and write to that section of memory, other applications can access that same section of memory, and the icing on the cake is that, unlike Critical Sections, the last application running shuts the lights off when it leaves, even after a GPF.

Implementing the Memory Mapped Files

Well, now that I had my solution I had to implement it. I did a quick search on MSDN and found the Win32API calls needed to work with Memory Mapped Files:

```
Declare Function OpenFileMapping Lib "kernel32"
  Alias "OpenFileMappingA"
  (ByVal dwDesiredAccess As Long,
  ByVal bInheritHandle As Long,
  ByVal lpName As String) As Long
Declare Function MapViewOfFile Lib "kernel32"
  (ByVal hFileMappingObject As Long,
  ByVal dwDesiredAccess As Long,
  ByVal dwFileOffsetHigh As Long,
  ByVal dwFileOffsetLow As Long,
  ByVal dwNumberOfBytesToMap As Long) As Long
Declare Function UnmapViewOfFile Lib "kernel32"
  (lpBaseAddress As Any) As Long
Declare Function CloseHandle Lib "kernel32"
  (ByVal hObject As Long) As Long
```

In Clarion the prototypes would be inside the Global Map and would look more like this:

```
  MODULE('')
    OpenFileMappingA(DWORD dwDesiredAccess, |
      BOOL bInheritHandle , ULONG lpName ) |
      ,HANDLE,PASCAL
    CreateFileMappingA(HANDLE hFile , |
      LONG lpAttributes , DWORD flProtect , |
      DWORD dwMaximumSizeHigh , DWORD dwMaximumSizeLow|
      , ULONG lpName ),HANDLE,PASCAL
    MapViewOfFile(HANDLE hFileMappingObject , DWORD dwDesiredAccess , |
      DWORD dwFileOffsetHigh , DWORD dwFileOffsetLow , |
      DWORD dwNumberOfBytesToMap),ULONG,PASCAL
    UnmapViewOfFile(ULONG hFileMappingObject ),BOOL,PASCAL
    FlushViewOfFile(ULONG hFileMappingObject , DWORD),BOOL,PASCAL
    memcpy(ULONG,ULONG,UNSIGNED),ULONG,PROC,NAME('_memcpy')
  END
```

OpenFileMapping

`OpenFileMapping` opens the memory map for reading, and returns a handle. Here are the parameters as described by Microsoft:

> ***dwDesiredAccess***
>
> *Access to the file mapping object. This access is checked against any security descriptor on the target file mapping object.*
>
> In other words, this parameter indicates what kind of access you want for the memory mapped file. I needed full access from all applications, so I used `FILE_MAP_ALL_ACCESS`. A list of the possible security descriptors is included in my sample code.
>
> ***bInheritHandle***
>
> *If this parameter is TRUE, the new process inherits the handle.*
>
> I don't need to inherit the handle for my use, so I set this to `FALSE`.
>
> ***lpName***
>
> *Pointer to a string that names the file mapping object to be opened. If there is an open handle to a file mapping object by this name and the security descriptor on the mapping object does not conflict with the dwDesiredAccess parameter, the open operation succeeds.*
>
> In other words, this is a pointer to a `CString` that contains the name of the Memory Mapped File.

Calling the API is very simple. I prime my `CString` with the name I want to give the Memory Mapped File, and make the call.

```
lCstrParam = 'MYLOGINFO'
hFileMap = OpenFileMappingA(FILE_MAP_ALL_ACCESS, FALSE, ADDRESS(lCstrParam))
```

CreateFileMapping

`CreateFileMapping` creates the memory map for reading, and returns a handle. Here are the parameters as described by Microsoft:

> ***hFile***
>
> *Handle to the file from which to create a mapping object.*
>
> *If hFile is INVALID_HANDLE_VALUE, the calling process must also specify a mapping object size in the dwMaximumSizeHigh and dwMaximumSizeLow parameters.*

lpAttributes

[in] Pointer to a SECURITY_ATTRIBUTES structure that determines whether the returned handle can be inherited by child processes. If lpAttributes is NULL, the handle cannot be inherited.

flProtect

Protection desired for the file view, when the file is mapped.

dwMaximumSizeHigh

High-order DWORD of the maximum size of the file mapping object.

dwMaximumSizeLow

Low-order DWORD of the maximum size of the file mapping object. If this parameter and dwMaximumSizeHigh are zero, the maximum size of the file mapping object is equal to the current size of the file identified by hFile.

An attempt to map a file with a length of zero in this manner fails with an error code of ERROR_FILE_INVALID. Applications should test for files with a length of zero and reject such files.

lpName

Pointer to a null-terminated string specifying the name of the mapping object.

If this parameter matches the name of an existing named mapping object, the function requests access to the mapping object with the protection specified by flProtect.

If this parameter is NULL, the mapping object is created without a name.

If lpNamematches the name of an existing event, semaphore, mutex, waitable timer, or job object, the function fails.

This call is pretty straightforward (see mmf.clw in the source zip for the definition of the equates):

```
lCstrParam = 'MYLOGINFO'
hFileMap = CreateFileMappingA(0FFFFFFFFh, 0, PAGE_READWRITE, 0, 1024,
ADDRESS(lCstrParam))
```

The application in the downloadable zip attempts to open a Memory Mapped File named MYLOGINFO, which will be created if it does not exist. `CreateFileMapping`, like `OpenFileMapping`, will return a handle to the file if successful. In this example, I've requested 1024 KB of memory for the file. `CreateFileMapping` will create the Memory Mapped File, `OpenFilemapping` will open a Memory Mapped File that already exists. I generally will attempt to open the file first, and if that fails I create the file.

Each program that you want to share this information will need to use MYLOGOINFO as the name of the Memory Mapped File. (Or any other name, as long as they all match) and each one of them will need to try to open the file first, and if it fails then create the file.

MapViewOfFile

`MapViewOfFile` maps a 'view' of the file, or the start of the address space. If `CreateFileMapping` or `OpenFileMapping` succeeds, `hFileMap` will contain a handle to the file. Next you have to initialize the memory by "mapping a view" of the file.

Here are the parameters for `MapViewOfFile`, as described by Microsoft:

hFileMappingObject

Handle to an open handle of a file mapping object. The CreateFileMapping and OpenFileMapping functions return this handle.

dwDesiredAccess

Type of access to the file mapping object, and therefore, the protection of the pages.

dwFileOffsetHigh

High-order DWORD of the file offset where the view is to begin.

dwFileOffsetLow

Low-order DWORD of the file offset where the view is to begin. The combination of the high and low offsets must specify an offset within the file that matches the system's memory allocation granularity, or the function fails. That is, the offset must be a multiple of the allocation granularity. Use the GetSystemInfo function, which fills in the members of a SYSTEM_INFO structure, to obtain the system's memory allocation granularity.

dwNumberOfBytesToMap

Number of bytes of the file mapping to map to the view. If this parameter is zero, the mapping extends from the specified offset to the end of the section.

Calling this API is simple enough also. You pass it the handle retrieved by either `CreateMapFile` or `OpenMapFile`, and the access level. It will return the starting address for your mapped file.

```
lpMapAddress = MapViewOfFile(hFileMap, FILE_MAP_ALL_ACCESS, 0, 0, 0)
```

The starting address and the size of the file are really the only things you need to read and write the file from this point on.

This one API call cost me hours of debugging time. After rolling this out to one client, who of course was one of my most detail-oriented, I noticed that there was a memory leak. I finally found the source of the leak when I realized it was growing by exactly 8KB per second. I was reading the file on a timer event, to be able to react to changes made by other applications. The size and timing of the leak coincided with the frequency and size of the Memory Mapped File access happening in the application. I learned not to Map something without Unmapping it, and that you really only need to Map and Unmap once. I'll go into more detail about this later. Another valuable lesson in paying attention to what you're doing when you're putzing around with Windows memory.

Reading the Mapped File

To read the mapped file I copy the data from the memory file into a locally declared data structure, using `MemCpy`. I won't go into a lot of detail on `MemCpy`, as there are plenty of resources on it, and the call is pretty straightforward.

```
MapGroup     GROUP,PRE(MG)
Login          CString(51)
Password       CString(26)
SecLevel       Long
             END
  CODE
  IF lpMapAddress
    memcpy(ADDRESS(MapGroup), lpMapAddress, SIZE(MapGroup))
  END
```

This is where I check to see whether another program has logged in. If the Memory Mapped File contains login data, and my program has not logged in yet, I bypass the login procedure and use the data from the Memory Mapped File. If my program is logged in already, and another program changes the login information, I need to change my local login information and security settings to match the new data from the Memory Mapped File. This is demonstrated in the sample application files included with this article.

Writing to the Mapped File

Writing to the Memory Mapped File is the exact opposite of reading it. You copy the information in your locally declared data structure back to the Memory Mapped File using `MemCpy`.

```
IF lpMapAddress
  memcpy(lpMapAddress, ADDRESS(MapGroup), SIZE(MapGroup))
END
```

Once a user logs in, this is the code used to put the login info back up for the other programs to read.

FlushViewOfFile

`FlushViewOfFile` unmaps the view when you're finished. If you look up FlushViewOfFile you will notice that Memory Mapped Files are not limited to being on one machine. You can actually do this across multiple machines on a network, allowing you to share data between programs on two different machines. That's a topic for another time and another article.

UnmapViewOfFile

UnmapViewOfFile unmaps the view when I'm finished. The parameters as described by Microsoft:

> ***hFileMappingObject***
>
> *Handle to an open handle of a file mapping object. The CreateFileMapping and OpenFileMapping functions return this handle.*

When I'm done with the memory-mapped file, I need to clean up after myself. Things get messy in a hurry, memory leak-wise, if I don't. Unmapping the file lets Windows know that I've finished and it can reuse the memory somewhere else if needed. Again, this API call is simple and straightforward.

```
IF ~UnmapViewOfFile(lpMapAddress)
  !Put Error Handling code here if needed.
END
```

Using a Memory Mapped File is really not that difficult. Create the file, map to the file, write to the file. Another application can open the file, map to the file, and read the file:

```
lCstrParam = 'MYLOGINFO'
hFileMap = OpenFileMappingA(FILE_MAP_ALL_ACCESS, |
  FALSE, ADDRESS(lCstrParam))
IF ~hFileMap
  hFileMap = CreateFileMappingA(0FFFFFFFFh, 0, |
    PAGE_READWRITE, 0, 1024, ADDRESS(lCstrParam))
END
IF hFileMap
  lpMapAddress = MapViewOfFile(hFileMap, FILE_MAP_ALL_ACCESS, 0, 0, 0)
END
IF lpMapAddress
  memcpy(ADDRESS(MapGroup), lpMapAddress, SIZE(MapGroup))
END
If MapGroup.Login and MapGroup.SecLevel
  ! Use the current Login and Security Level
  Loc:Login = Mapgroup.Login
  Loc:Password = MapGroup.Password
  Loc:SecLevel = MapGroup.SecLevel
Else
  Do LoginRoutine
  IF lpMapAddress
    memcpy(lpMapAddress, ADDRESS(MapGroup), SIZE(MapGroup))
  END
```

```
END
IF ~UnmapViewOfFile(lpMapAddress)
  !Put Error Handling code here if needed.
END
```

When using Memory Mapped Files to keep two applications in sync, you only need to map the file once after opening it and unmap it once when you're done with it.

Both applications can read or write to the file as long as they have the correct permission levels declared, and both need to unmap when they're finished. When the last application using the file ends, the memory that the file was using is reallocated by Windows. And the best part is that there are no residual files hanging around to be read by inquisitive eyes.

Summary

Memory Mapped Files can be used for many purposes other than the one I presented here. For instance, you can control other applications via values in the Memory Mapped File. You can even have programs that monitor the existence of the Memory Mapped File and perform tasks only when the file has been created. Your creativity is all that holds you back.

Something you should keep in mind when using Memory Mapped Files is that the memory allocated is contiguously in 8 KB blocks, meaning that if you need 17 KB of space, 24 KB of space will be allocated. If Windows cannot allocate the amount of memory requested due to memory being low or due to fragmentation of the memory, the call to MapViewofFile will fail. Fragmentation of the memory is handled a little more efficiently in Windows XP and Windows 2003 with the introduction of the Low-fragmentation Heap. Windows XP Low-fragmentation Heap can also be downloaded for Windows 2000 from Microsoft.

But even with the cost of memory coming down and Windows handling its heap memory better, it's still best to use Memory Mapped Files sparingly. Don't try to declare a file to hold your entire personnel database; instead, declare just enough room to pass a single record. Or even better, just pass the pertinent information needed to complete your task. Memory is cheap these days, but you never know what type of setup your program is going to be running on. Always plan for the worst, and hope for the best.

Download the source

Marty Honea wrote his first program at a summer camp when he was 9, in BASIC on an Apple IIe. Among other jobs, he has worked as a draftsman, as an Emergency Medical Technician, and as an IT manager for three prisons. He picked up Clarion for Windows in 1998 and, despite having no formal training in computer programming, has managed to make a living with Clarion ever since.

Reader Comments

Add a comment

- » Great article Marty... thanks for your insights!
- » Thanks Ben.
- » You'll want to give careful consideration to naming your...
- » Thanks Carl, The name might be a bad Idea. I...

Clarion Magazine

.NET Basics: What Is .NET, And Why Should I Care? Part 2

by David Harms

Published 2005-11-24

In Part 1 of this series I gave a ridiculously brief summary of the history of Windows programming, from Windows API to COM and Active X, to the .NET framework. I also discussed some basic concepts such as the .NET framework, IL code, and namespaces. In this installment I'll take a closer (but still pretty high level) look at the classes that make up the .NET framework. And to do that I need to explain a bit more about namespaces.

Namespaces

As I indicated in Part 1, namespaces are a way of dividing classes up into named groups, both to make it easier to find what you're looking for, and to avoid name collisions.

In Clarion as we now know it, there is no formal definition of namespaces. In one sense, every Clarion application is a single namespace. For instance, you cannot define two procedures with the same prototype in one application, and you cannot have two global variables or classes with the same name. You *can* create your own functions with the same name and prototype as runtime library functions, but you will get a "redefining system intrinsic" message.

On the other hand, two different classes can each implement methods with the exact same signature, and two different procedures can each declare identical local variables. So classes and procedures, in a sense, have their own namespace, and the names used therein will not conflict with identical names in other classes or procedures.

At best, however, this is a two or three level hierarchy, which is not all that useful. If you were to create a library of all the reusable, shareable classes and procedures written by all Clarion programmers, there would be an incredible amount of name duplication.

In .NET, however, every class must belong to a specific namespace. You decide the namespace's name, and it can be anything you want. The convention, however, is for the top level namespace to be your company name or your domain name.

Namespace hierarchies

Namespaces are hierarchical. That is, you don't have just a single set of namespaces at the top; rather, each namespace can be further divided into sub-namespaces. The top level namespace in the .NET framework is `System`, which contains many widely used classes (including base data types) and interfaces. System also includes the `Object` class,

which is the topmost class in the hierarchy - if you create a .NET class not derived from any other class, it is still implicitly derived from `Object`. Most of Microsoft's .NET framework namespaces are descended from System. For instance, there is a `System.Text` namespace, for general string handling functions, as well as a `System.Text.RegularExpressions` namespace for just regular expression classes. The deeper you go in the hierarchy, the more specific the namespaces become.

In Java (from which C# is, at least in some ways, descended), namespaces are called packages, and the convention is to name your package after a domain name, with the elements reversed. For instance, all of the server-side Java code I've written for the Clarion Magazine server is organized under com.covecomm (e.g. `com.covecomm.webmag.control`). I like the Java approach: domain names are globally unique, so you're guaranteed not to have name clashes. Microsoft's approach leaves the door open for conflicts (imagine Acme Web Widgets Inc. and Acme Worldwide Web Programming Inc. both creating, and making public an `acme.web namespace`) but .NET namespaces are still a huge improvement over what is essentially one big namespace in Win32.

Another important difference between Java packages and .NET namespaces is in the physical storage of the source code. In Java, class source files reside in a directory structure that corresponds to the package. My ControllerServlet class, which is in the `com.covecomm.webmag.control` package, has a fully qualified file name like `src\com\covecomm\webmag\control\ControllerServlet.java`. In .NET you are under no such restrictions – you can put your source code anywhere you like, and you can even have multiple namespaces in a single source file.

The .NET framework namespaces

While Microsoft has only released .NET code using two top-level namespaces, Microsoft and System, the entire .NET framework namespace hierarchy contains over a hundred namespaces. Here's an annotated list (as of Nov 2005):

`Microsoft.CSharp`
`Microsoft.JScript`
`Microsoft.VisualBasic`
`Microsoft.Vsa`
	Classes to support specific MS languages on .NET

`Microsoft.Win32`	Win32 registry and event handling classes
`System`	The very basic classes on which .NET is built.
`System.CodeDom` `System.CodeDom.Compiler`	Classes to support the addition of new .NET languages – SV is making use of this stuff in Clarion.NET
`System.Collections` `System.Collections.Specialized`	Queue-like operations.

`System.ComponentModel`
`System.ComponentModel.Design`
`System.ComponentModel.Design↵`
`.Serialization`
	Managing components and controls. The well-defined component model means, among other things, that the Clarion.NET window formatter will be able to use virtually any .NET control. In the current Win32 formatter, adding support for a new control means modifying and recompiling the formatter, which only SV can do. Serialization is the process of converting an object into a storable form, and vice versa.

System.Configuration System.Configuration.Assemblies System.Configuration.Install	.NET configuration stuff, including writing installers
System.Data System.Data.Common System.Data.Odbc System.Data.OleDb System.Data.OracleClient System.Data.SqlClient System.Data.SqlServerCE System.Data.SqlTypes	Working with data sources, including ADO.NET
System.Diagnostics System.Diagnostics.SymbolStore	Tracing and debugging
System.DirectoryServices	Working with Active Directory
System.Drawing System.Drawing.Design System.Drawing.Drawing2D System.Drawing.Imaging System.Drawing.Printing System.Drawing.Text	All that good graphics stuff you used to have to do with the GDI.
System.EnterpriseServices System.EnterpriseServices↵ .CompensatingResourceManager System.EnterpriseServices.Internal	Using COM+ from within .NET
System.Globalization	Internationalizing applications – time and date formats, string sorting, etc.
System.IO System.IO.IsolatedStorage	Reading/writing data streams and files
System.Management System.Management.Instrumentation	Expose your application to Windows Management Instrumentation, so your app can be more easily managed with MS tools.
System.Messaging	Send and receive messages on the network
System.Net System.Net.Sockets	Handle network protocols, including HTTP (lots of web-related classes here) and winsock.
System.Reflection System.Reflection.Emit	Reflection makes it possible for one class to inspect another class and discover its methods and properties. A vast improvement on the primitive form available in COM.
System.Resources	Manage resource bundles, e.g. for translation
System.Runtime.CompilerServices	For compiler writers only
System.Runtime.InteropServices System.Runtime.InteropServices↵ .CustomMarshalers System.Runtime.InteropServices↵ .Expando	Interoperability with other languages. See Wade Hatler's excellent series for examples of using these classes with Clarion.

System.Runtime.Remoting System.Runtime.Remoting.Activation System.Runtime.Remoting.Channels System.Runtime.Remoting.Channels.Http System.Runtime.Remoting.Channels.Tcp System.Runtime.Remoting.Contexts System.Runtime.Remoting.Lifetime System.Runtime.Remoting.Messaging System.Runtime.Remoting.Metadata System.Runtime.Remoting.Metadata↵ .W3cXsd2001 System.Runtime.Remoting.MetadataServices System.Runtime.Remoting.Proxies System.Runtime.Remoting.Services	Classes to support software as a service, SOAP, and a whole lot more.
System.Runtime.Serialization System.Runtime.Serialization↵ .Formatters System.Runtime.Serialization↵ .Formatters.Binary System.Runtime.Serialization↵ .Formatters.Soap	Serialization is the process of converting an object or set of objects into simple storage (e.g. in a database), and restoring the object(s) from the data at a later time.
System.Security System.Security.Cryptography System.Security.Cryptography↵ .X509Certificates System.Security.Cryptography.Xml System.Security.Permissions System.Security.Policy System.Security.Principal	Security tools for applications, including managing what sorts of things an application may do (e.g. may read/write files, may only read files, etc). Also includes cryptography services.
System.ServiceProcess	Creating a Windows service
System.Text System.Text.RegularExpressions	Manipulating string data, regular expressions
System.Threading	Managing threads
System.Timers	Managing timers
System.Web System.Web.Caching System.Web.Configuration System.Web.Hosting System.Web.Mail System.Web.Mobile System.Web.Security System.Web.Services System.Web.Services.Configuration System.Web.Services.Description System.Web.Services.Discovery System.Web.Services.Protocols System.Web.SessionState System.Web.UI System.Web.UI.Design System.Web.UI.Design.WebControls System.Web.UI.HtmlControls System.Web.UI.MobileControls System.Web.UI.MobileControls↵ .Adapters System.Web.UI.WebControls	ASP.NET stuff, including IIS-independent hosting, sending email, mobile web apps, web services and more SOAP, HTML, web, and mobile controls, etc.

`System.Windows.Forms` `System.Windows.Forms.Design`	The big one for Clarion developers – WinForms is the basis for all Windows forms and controls in .NET. In Clarion.NET the WINDOW structure will (at least initially) be replaced by a WinForms window definition.

`System.Xml`
`System.Xml.Schema`
`System.Xml.Serialization`
`System.Xml.XPath`
`System.Xml.Xsl`
	XML support, including saving/restoring, searching, and formatting.

These are the core .NET framework classes which are guaranteed to be present on any Windows machine with .NET installed. Well, mostly – the above list does include some version 1.1 only namespaces, and there are also some deprecated methods in .NET 2.0. Which raises another interesting point - which version(s) of .NET do you want, and which version(s) will Clarion.NET support?

Which version of .NET do I want?

Version 2.0 of the framework is still pretty new, but I expect that most .NET development will move to 2.0 fairly quickly. Right now the only compelling reason I'm aware of to stick with 1.1 is for Mono compatibility.

Mono is the Novell-sponsored effort to create an open source implementation of .NET that will run on Unix/Linux. Naturally, Mono releases lag behind .NET releases, and right now Mono 1.1 is without WinForms support. So right now, even if you had Clarion.NET in your hot little hands, you'd be restricted to .NET 1.1 apps that do things that don't need desktop forms, like ASP.NET. WinForms is expected in Mono 1.2, which is still a .NET 1.1 framework. The idea would be for Clarion.NET to create .NET 2.0 apps and Mono 2.0 to be in a usable form. Mono developers are not even talking dates for 2.0 yet, which suggests to me that there's some major work still to be done on the 1.2 product. I wouldn't sign any contracts for Linux apps just yet.

The initial, hand coder's release of Clarion.NET will probably run on, and target, .NET 1.1, although there's a possibility it may create .NET 2.0 apps. By release, I expect Clarion.NET to run on 2.0 and hopefully target either 1.1 or 2.0 apps. By all accounts it's actually much easier to create apps for 2.0 than 1.1.

Why should I care?

Okay, so now you know a little about the .NET framework library. What good does that do you? Lots! If you've ever worked with the Windows API, you know what a hodgepodge of function calls that is. Half the difficulty is just finding out which API call you want to use. The .NET framework library, on the other hand, is neatly organized into namespaces, making it much easier to find what you're looking for, particularly with the handy dandy annotated table above.

And once you find the part of the library you want, you're dealing with classes, not with function calls. One class can replace a whole bunch of function calls, making code a lot cleaner and easier to maintain.

For example, the NET Framework SDK 2.0 Beta 2 Samples download includes a project called `WordCount` which, as the name suggests, counts words in a document. You can easily incorporate `WordCount`'s functionality into your own .NET apps, but as written it runs as a command line utility:

```
Usage: WordCount [-a] [-o] [-f<output-pathname>] [-c<codepage>] input-pathname...
```

```
   -? Show this usage information
   -a Word usage sorted alphabetically
   -o Word usage sorted by occurrence and then alphabetically
   -f Send output to specified pathname instead of the console
   -c Specify the codepage to use to read the file
```

There are actually several classes in the `WordCount` project, including a nifty reusable command line argument parser. The code I want to look at, however is contained in the `WordCounter.CountStats` method (I've left out some bits before, during, and after). This is from the C# version:

```
// Attempt to open the input file for read-only access
FileStream fsIn = new FileStream(pathname, FileMode.Open,
                    FileAccess.Read, FileShare.Read);

numBytes = fsIn.Length;
using (StreamReader sr = new StreamReader(fsIn, fileEncoding, true))
{
   // Process every line in the file
   for (String Line = sr.ReadLine(); Line != null; Line = sr.ReadLine())
   {
      numLines++;
      numChars += Line.Length;

      String[] Words = Line.Split(null);  // Split the line into words

      for (int Word = 0; Word < Words.Length; Word++)
      {
         if (Words[Word].Length > 0)
         {   // Don't count empty strings
            numWords++;
                     // some stuff omitted that counts word instances
         }
      }
   }
}
```

If you're not familiar with C#, just mentally strip out the semicolons at the end of each line and ignore most of the braces, and the code will look a lot like Clarion.

First, think about how you'd accomplish the task of counting words using Clarion. First, you'd declare an ASCII file, then you'd set up a loop to read through the file one record at a time. You'd retrieve each record, and then you'd have to write some code to parse each record into words.

Here's how the `WordCount` example uses the .NET framework library for these tasks. First, the code reads the file using combination of a `System.IO.FileStream` object (called `fsIn`) to represent the file, and a `System.IO.StreamReader` object to read the file one line at a time. The `using` directive tells C# to dispose of the sr object when the following code block (enclosed in `{}`) completes:

```
using (StreamReader sr = new StreamReader(fsIn, fileEncoding, true))
```

The first `for` loop gets one line at a time from the file:

```
for (String Line = sr.ReadLine(); Line != null; Line = sr.ReadLine())
```

Now comes a cool bit of code, involving an array of strings (`Words`) and a single string containing one line of the text file (`Line`):

```
String[] Words = Line.Split(null);
```

All strings in .NET are instances of the `System.String` class, and `System.String` has a ton of methods. In Clarion, you use library functions like `CLIP` and `LEFT` to manpulate strings; in .NET, you use the string's own methods. The `Split` method returns an array of the substrings that make up the string. By default, the delimiter is a space. If, for instance, you had a file of comma-delimited words with no spaces, `Line.Split(null)` would return each line as a single word. You can specify up to six additional delimiters as char parameters to `Split`; with comma and period separators, the line would look like this (single character, or char, constants are enclosed in single quotes, rather than the double quotes used for string constants):

```
String[] Words = Line.Split(',','.');
```

The second `for` loop iterates through the words in the array so they can be counted:

```
for (int Word = 0; Word < Words.Length; Word++)
```

I've omitted some other bits of code, such as the tests to see whether a specific word has already been counted.

If I already had Clarion.NET, then I could easily incorporate the `WordCounter` class into my own apps. The declaration would look something like this:

```
wc            WordCount
fileName      string
fileEncoding  string

numLines      long
numWords      long
numChars      long
numBytes      long
```

And the code (with the numeric variables passed by address):

```
wc.CountStats(fileName,fileEncoding,numLines,numWords,numChars,numBytes)
```

I'd probably also need to include the `WordCount` assembly in my project, and I might want a `using Microsoft.Samples` statement somewhere so I wouldn't have to refer to `WordCount` as `Microsoft.Samples.WordCount`.

If you look at the `WordCounter` source, remember that C# is case sensitive, and `WordCounter`, which is the class that manages the overall counting, is not the same thing as `wordCounter`, which is an instance of the `System.Collections.SortedList` class, and is declared as a member of `WordCounter`. That's a sloppy bit of naming, in my opinion - `wordCounter` should probably have been something like `wordList` to avoid confusion. But hey, it's just an example. And I do suggest you take a look at the `System.Collections` namespace. In Clarion we've been quite spoiled with `QUEUE`s, and the `Collections` classes certainly aren't a replacement, but they have some very nice features. I'll be interested to see what Clarion.NET does with queues...

This example barely scratches the surface of what you can do with the framework classes. The key points to remember are:

- It's a lot easier to find classes for a specific task because everything is organized into namespaces.
- Classes represent much more complete functionality than individual API calls
- In any .NET language, you can use the framework classes (and all other .NET classes) natively - no funny prototyping or mangling required.

Beyond the .NET framework

So far I've looked at just the .NET framework namespaces. There are certainly a lot of extremely useful classes there, with more to come from Microsoft (see my blog entry on LINQ), as well as many code samples showing how to work with the framework classes.

Using the .NET framework is just the beginning, however. Just as there are many third party products you can use with Clarion, so there are numerous third party products you can use with any .NET language, including Clarion.NET. But that's a subject for 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 produces the Planet Clarion podcast, which he co-hosts with Andrew Guidroz II.

Reader Comments

Add a comment

- » Coding in Clarion and porting to Linux and Mac is closer...

Clarion Magazine

A FileDropBox With Conditional Content

by Maarten Veenstra

Published 2005-11-30

There are always situations where the contents of a FileDropBox, either a FileDropListBox or a FileDropCombo, is dependant on some criteria, usually the contents of an entry field or another FileDropBox. But a FileDropBox is always file loaded, and this loading is done when the form opens; after that the FileDropBox remains pretty static. In this article I'll show you how to change the FileDropBox's content at runtime.

Setting up the FileDropBox

I always use local variables as the `USE()` variable for the FileDropBox. I have made my own convention to use the prefix `FDL:` for a FileDropListBox and `FDC:` for a FileDropComboBox, followed with the label of the field they represent. I declare the variables using the Derived From option - this will keep them in sync with the DCT, and makes them easily recognisable in the embed tree. Also, by using local variables, I don't need alias files when I have more than one FileDropBox on the same table. When I do need to display fields from this table, then of course I'll need local variables for those as well (with multiple FDBs).

Refreshing the FileDropBox

In the downloadable source at the end of this article there is a simple example of conditional content for a FileDropBox. This example uses the `Street` and `City` tables, and the records in the `Street` FileDropBox are dependent on the selection from the `City` FileDropBox. This situation is easy to solve. Assume you have an `Address` table containing a list of pointers to address elements. `ADR:CTYid` is a (`Long`) pointer to a record in the `City` table, and `ADR:STRid` is a (`Long`) pointer to a record in the `Streets` table. Declare three local variables: `FDC:City Like(CIT:City)`, `FDC:Street Like(STR:Street)` and `MEM:CTYid`.

I'm using `Like()` to indicate I derived the local variables from their matching file structure elements.

Populate the City-FileDropCombo on the window and use `FDC:City` for its `USE()` variable. On the Actions tab, set Field to fill from to `CTY:CTYid`, and Target field to `ADR:CTYid` (see Figure 1).

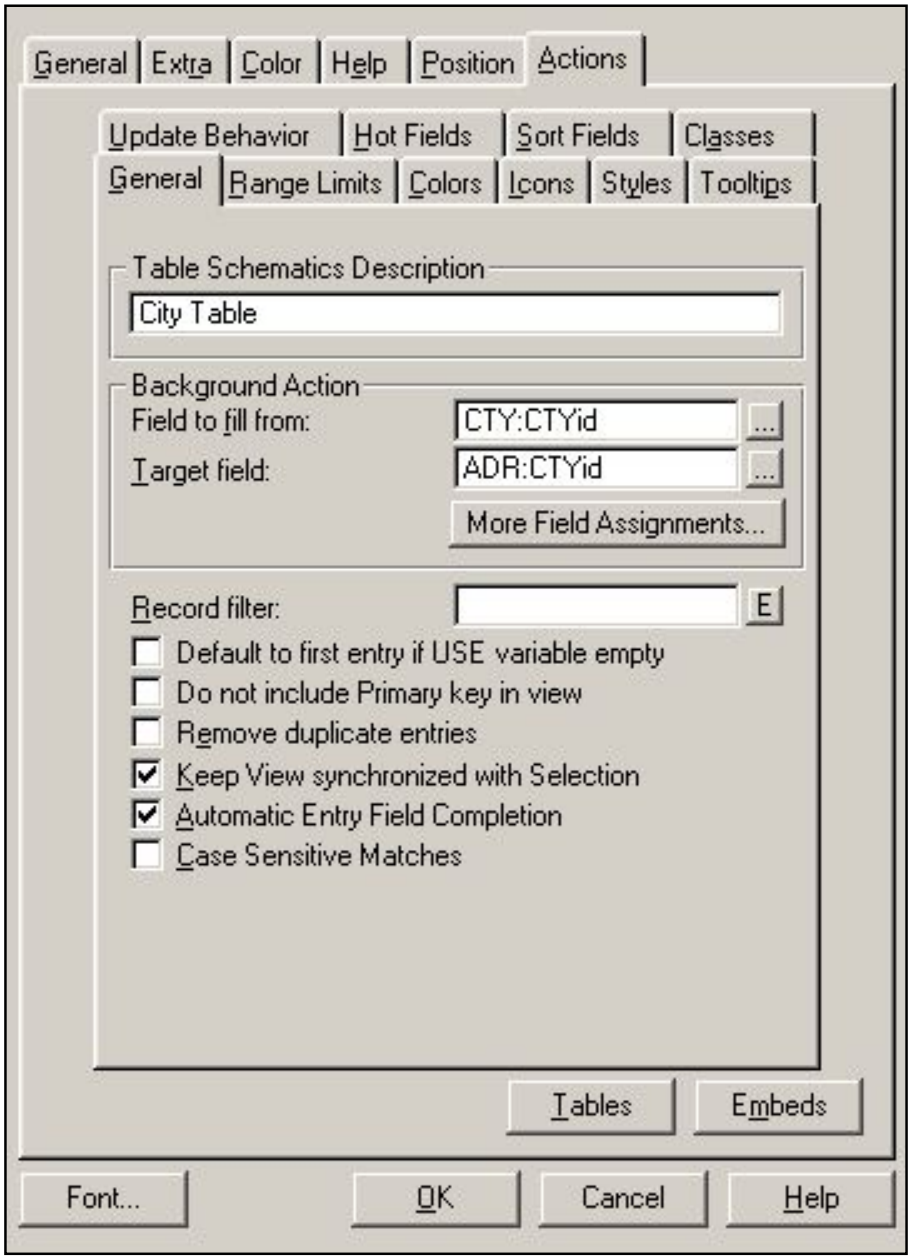


Figure 1. City table settings

Populate the Streets-FileDropCombo on the window and use `FDC:Street` for its `USE()` variable; on the Actions tab, set Field to fill from to `STR:STRid` and Target field to `ADR:STRid`. Then, on the Range Limits tab, set Range Limit Field to `STR:CTYid`, set the Range Limit Type to Single Value, and set Range Limit Value to `ADR:CTYid` (see Figure 2).

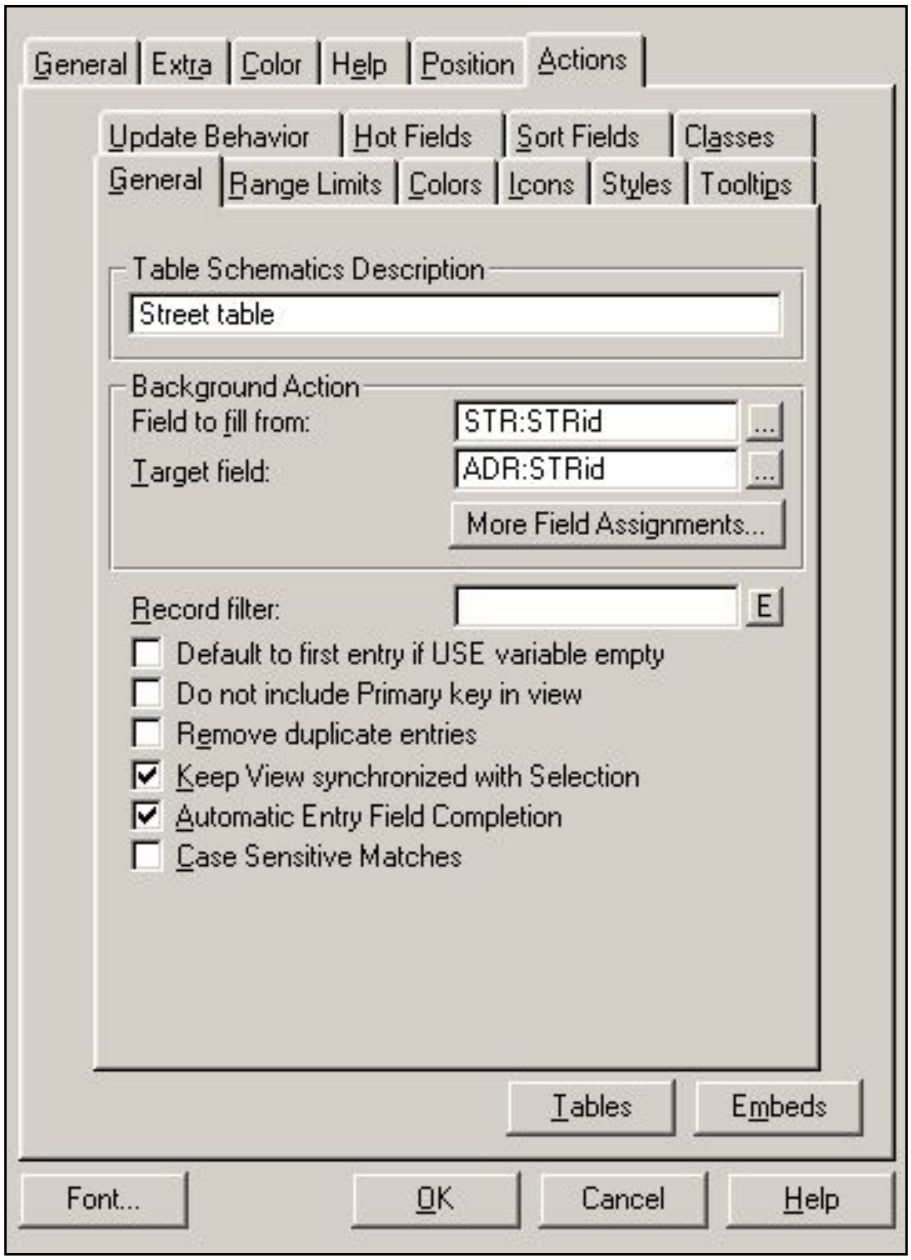


Figure 2. Street FileDropCombo settings

Then in the embed: WindowManager, Init, After Opening Files, enter:

```
MEM:CTYid = ADR:CTYid.
```

Since a FileDropBox has no reset fields, you'll have to do the reset yourself. In the Event:Accepted for `?FDC:City` enter:

```
IF ADR:CTYid <> MEM:CTYid
  FDCB2.ResetQueue(True)   !=Streets-FDC
  MEM:CTYid = ADR:CTYid
END
```

And you're done! The ResetQueue will refill the list box with the new data.

A FileDropBox with a different source (file)

You'll have a more complex situation when the contents of a FileDropBox is not simply dependent on its filter, but needs to come out of an entirely different table when conditions change! You can, of course, populate two FileDropBoxes on top of each other and hide one or the other, depending on the criteria. Or you can create a manual filter based on the common dominator….

Here's a nice, complex situation. In my app, the end user can put all components used (ignoring for the moment what these components are) in one table, which is used as a pool for the components used in a project:

```
Pool-Table      ProjectPool-Table      Project-Table
POOLid     <->> POOLid
                PROJid            <<-> PROJid
```

Above is a typical relation scheme where the `ProjectPool` table is used for a many to many relationship between the `Project` table and the `Pool` table. When there are components used in a `Project`, I set up a FileDropBox on the `ProjectPool` table, limited on PROJid, to show only the components from the `Pool` table that are used in that `Project`.

Within the `Project` I have an `Object` table, which uses components from the `Pool` table. But not all components from the `Project`/`Pool` can be used on every `Object`. For instance, you can't turn an English bolt into a metric nut, where both English and metric bolts & nuts are in the `Pool` table. To guard the end user from making such a mistake, an optional many to many table is added between the `Objects` table and the `Pool` table:

```
Pool-Table      ObjectPool-Table      Object-Table
POOLid     <->> POOLid
                OBJid             <<-> OBJid
```

Easy enough you'd say: just point the FileDropBox to the `ObjectPool` table, limited on `OBJid`. That's true if the `ObjectPool` table wasn't optional! Because it is optional I need to use the `ObjectPool` table when there are any items entered for that specific `Object`, else I need to use the `ProjectPool` table.

The common dominator in both situations is the `Pool` table. This means that I setup the `FileDropBox` to use the `Pool` table and go for manual filtering. To speed up things, I decided to load both the `ObjectPool` table and the `ProjectPool` table into a queue.

```
OBJque  QUEUE,PRE(OQ) !Queue for filtering Component of one Object
POOLid    LONG        !Pointer to the Pool table
        END
OBJqueRecs  LONG      !Number of records in the OBJque
PROJque QUEUE,PRE(PQ) !Queue for filtering Component of one Project
POOLid    LONG        !Pointer to the Pool table
        END
```

The `PROJque` data doesn't change while the user is on the form, so this can be filled in the Form's WindowManager.Init, after opening the files:

```
ProjectPool.PROJid = Project.PROJid
Set(ProjectPoolPROJid_Key, ProjectPoolPROJid_Key)
Loop until Access:ProjectPool.Next()
  If ProjectPool.PROJid <> Project.PROJid
    Break
  End
  PQ:POOLid = ProjectPool.POOLid
  Add(PROJque, +PQ:POOLid)
End
```

The `OBJque` however is dependant on the chosen Object so the filling code is put in a ROUTINE:

```
LoadOBJque   ROUTINE
  Free(OBJque)            !Start fresh
  OBJqueRecs = 0         !There are no records
  !
  ObjectPool.OBJid = Object.OBJid
  Set(ObjectPoolOBJid_Key, ObjectPoolOBJid_Key)
  Loop until Access:ObjectPool.Next()
    If ObjectPool.OBJid <> Object.OBJid
      Break
    End
    OQ:POOLid = ObjectPool.POOLid
    Add(OBJque, +OQ:POOLid)
  End
  OBJqueRecs = Records(OBJque)
  !
  EXIT
```

Now this routine is called from `WindowManager.Init` and whenever the chosen `Object` changes.

Finally I need to embed the manual filter in the FileDropBox's `ValidateRecord` method,

before the parent:

```
IF OBJqueRecs  !We have Components specified for this Object
  OQ:POOLid = Pool.POOLid
  Get(OBJque, +OQ:POOLid)
  IF ErrorCode()            !Not found
    RETURN Record:Filtered
  END
ELSE           !Take from the current ProjectPool
  PQ:POOLid = Pool.POOLid
  Get(PROJque, +PQ:POOLid)
  IF ErrorCode()            !Not Found
    RETURN Record:Filtered
  END
END
```

With this manual filter I load the `Pool` table and filter it conditionally on either the `OBJque` (from the `ObjectPool` table) when there are entries found, or on the `PROJque` (from the `ProjectPool` table).

When, on the form, the chosen `Object` changes, I call the `LoadOBJque` routine and refresh the `FileDropBox` with a call to its `ResetQueue`, as shown above.

Summary

When you have FileDropBoxes with conditional data, it's easy to reload them at runtime using local variables and a few lines of embedded code. You can even extend this technique to use manual filtering to choose between two sources for a FileDropBox.

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

Add a comment

Clarion Magazine

Clarion Developer Bio: Sim Sherer

Published 2005-11-30

From 2001 to 2003, Sue Pichotta ran a series of Clarion developer bios for the IceTips News Network. You can find those bios archived here at Clarion Magazine. I've been meaning for some time to pick up the torch; hopefully Sim's bio will be the first of many.

Dave Harms

Who do you work for?

A documentation contracting house.

What do you like best about what you do now?

Translating the technical world to the uninitiated user world. Providing end-user online help and printed documentation.

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

Learning to use Clarion for Windows, especially with Hebrew (right-to-left).

What has been one of your biggest challenges in business?

Getting older people to use a computer for what is designed, i.e. *not* games.

Do you use any computer languages besides Clarion?

Not today.

When did you begin working with computers?

My computer life actually started of back in 1968. As a junior accountant, I grew tired of auditing and found that large clients could produce analysis reports quicker that I could work out what I wanted from them. This really got me going and I joined IBM in west London as a trainee programmer. I started with Basic Assembler Language (BAL), Report Programming Generator (RPG) and COBOL, using punch cards with DOS and TOS. TOS was tossed and DOS grew to DOS/VSE. This was followed by OS/VS1 and OS/VS2 which later became OS/MVS which still lives today, but I digress. I was involved with all this on mainframe computers.

In 1972 I left for sunny South Africa an rejoined IBM as a maintenance programmer. Two years later I built a colour tevision set, the first one that was home-built in South Africa, even the national press came to interview me and published my one and only press picture! After three years I joined NCR and eventually headed a team of eight programmers. I married and decided to get adventurous. I left NCR to join a software house that prided itself in a payroll system that was as cross-platform as was possible in those days. They were also the agents for Computer Associates financial software products. The office was about five minutes from my house which was a major playing card. I provided technical support, training and add-on applications, all in COBOL. When the US started embargoes against South Africa due to the existing regime, the financial products division was forced to shut shop. I continued to provide support and write add-on software in a private capacity.

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

In the course of events, I was approached by a colleague, a budding industrial psychologist, to write a PC application based on his thesis. I had never written PC application software in my life, I knew nothing of industrial psychology, nor had I ever really read a thesis that I could honestly say I had understood. This was a challenge!

The project revolved around matching people's personalities to their jobs. The idea was to allow a prospective employer to match an applicant's personal assessment of themselves with the expectations of a prospective employer. The requirement was to produce an EXE file (bug-free of course). After much running around I settled on Clarion as the development tool to use, CPD version 2.107. Almost three months later I had completed my learning curves AND produced the working program! My colleague completed his doctorate with honours too!

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

Yes, documentation. Due to the collapse of the Soviet regime, thousands of Russians came to Israel. The programming market was way over-supplied in all areas bar embedded architecture which really is not my field, so I embarked on technical writing. I had written instruction documents in South Africa and here in Israel, programmers with an electronics background and an English mother tongue are not as plentiful. Technical documentation was chosen as the winner.

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

Swimming and walking.

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

Married, 14 children of which four are married. Nine grandchildren plus two on the way. My siblings live in Manchester, England.

Where were you born?

I was born and bred in rainy Manchester in England. My late father was very ill with pneumonia during the war and was advised to move to a warmer and drier climate. In 1959 we move to Upington between the Gemsbok and the Kalahari desert. If it weren't for the Orange river, there would be no "between". Upington had only two tar roads, the

remainder were all sand and stones. The sand dunes were five minutes away by car. We lived there for almost one year and never saw any real rain! In 1960 we all moved to Johannesburg, and two years later I left for college in the USA. After six years I returned to England and settled in London (less rain than Manchester!).

Where do you live now?

In 1992 we (my family) decided that the South African political climate was getting too hot and started preparing for emigration to Israel, which is where we are today.

What's interesting about where you live?

No other Clarion programmers around to discuss concepts, ideas and accomplishments. (Is THAT interesting?)

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

I cannot really pinpoint one particular person.

Any other comments?

Programming is in my blood. My wife is often claims to be a computer widow! I went and invested in CW, version 2.003 in those days. Today I have a few application "users" about although I am now at C5.5 but I have not yet got into ABC. I call them users rather than customers because they are charity and non-profit organizations such as an orphanage, two hospices, and an old-age home etc. I don’t charge them and in turn hope I will never need their services.

Reader Comments

Add a comment

- » Greetings from Sunny South Africa. To put things into...

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

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

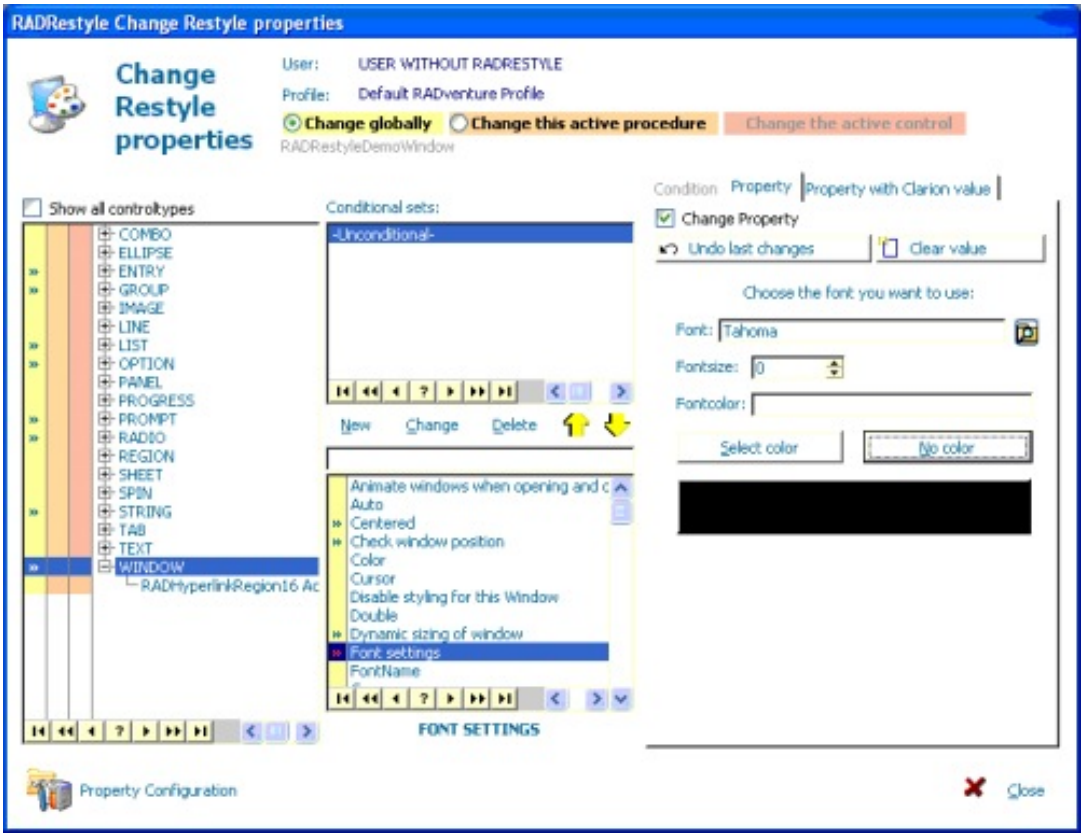
Rad Runtime Restyling (updated)

Direct link

Posted Wednesday, November 02, 2005 by Dave Harms

Last week RADVenture's Peter Rakke gave me a demo of his RADRestyle product. We talked over Skype, and Peter demonstrated the software using UltraVNC. RADRestyle is a tool that lets you apply run-time formatting changes to your Clarion application. These changes include just about anything you can do with the property syntax, from fonts and colors and positioning, and even goes as far as some kinds of field validation. As all of this has the potential to change not just how the program looks, but how it functions, so RADRestyle also includes security features so you can decide which users have access to what functionality.

One of the things I really like about RADRestyle is the Change Restyle properties form (below). Notice the color coded bars down the left side of the control types tree list. Now take a look at the same-color radio buttons at the top of the form. This nice visual layout makes it easy to see how your changes are affecting the program at the control, procedure, and application level.



As the screen shot suggests, RADRestyle is a complex product, and I'm really not even touching the surface of what it can do. Definitely take a look at the instructions page to get a better idea of RADRestyle's power.

So what's the catch? RADRestyle was developed for in-house use, and not specifically as a third party product, although several developers are using it already. Peter is evaluating whether to go the third party route, so if you're interested in RADRestyle email Peter at peter.rakke at adventure.nl

While Peter was doing the demo I noticed something peculiar about how he navigated filedialogs. Instead of pointing and clicking to get to a particular directory, Peter frequently typed in the name of a shortcut residing in the root directory. For instance, he might have a shortcut called xC6Libsrc pointing to C:\Clarion6\libsrc, and by typing in /xC6Libsrc (or portion thereof) and pressing Enter, he quickly moved to that directory. This is a very neat trick, and I've started doing the same, although with one change. I created a root directory called, simply, z, and to get to my Clarion 6 libsrc directory from any file dialog I type \z\ followed by the name of the shortcut, or I just select the shortcut from the list.

Update: I forgot to mention that RADVenture is the company behind Fenix, a Clarion template set that generates C# or VB.NET code for ASP.NET applications. Clarion Magazine recently published a review of Fenix.

Z turns up the Clarion.NET heat

Direct link

Posted Monday, November 07, 2005 by Dave Harms

Bob Zaunere has a blog entry titled What does Clarion.NET *really* offer over C#?. Z's points:

- a familiar runtime library designed for database apps
- a language designed for database apps
- and, of course, code generation

Z specifically says that Clarion.NET has "the .NET version of the Clarion runtime library (RTL), which means it has the familiar built-in data access methods, the Clarion database Drivers, the View Engine, the Report engine, Queues, Groups, and everything else that defines Clarion."

Certainly rewriting the RTL and the database drivers has to rank as one of the major obstacles to a full-fledged Clarion.NET. I'm hopeful that Z's post and the newly created Clarion.NET web forum are signs that a hand-coder's release is coming soon.

Z acknowledges that other vendors are encroaching on Clarion's code generation territory (emphasis added):

> One final comment, there are other code gen products out there today, and *still others coming* that will do their very best to mimic Clarion. But don't be fooled, twenty years experience in Code Gen technology is not something you pickup at the local grocer.

Competition on the code gen front is a mixed blessing. It will close the productivity gap. But it also helps validate Clarion's template approach. And I tend to agree that Clarion's long history with the technology is a strong advantage.

Clarion.NET hand coder's beta in sight?

Direct link

Posted Monday, November 07, 2005 by Dave Harms

Bob Z has posted a message reiterating that the first release of Clarion.NET will be a hand-coder's edition. Besides the lack of an AppGen, this release will have a few restrictions:

- limited release - it will only be available to selected developers who will give feedback to SV, and who have a "high pain threshold :-)". In other words, the first release will not be a public beta
- release date - possibly (remember, *possibly*) late Dec, or after the holidays.
- the initial release will use .NET 1.1 - the port to 2.0 is under way, but not complete. The final release of Clarion.NET will use .NET 2.0.

There's definitely a ton of interest in Clarion.NET, and it's good to see signs that the first release is close.

Clarion.NET data access options

Direct link

Posted Tuesday, November 08, 2005 by Dave Harms

Bob Z has outlined three ways you can use Clarion.NET to access databases:

- Standard Clarion syntax (NEXT, PREVIOUS, etc) - under the hood, these result in Win32 calls (unmanaged code) to the Clarion RTL. This is what gets used (at least for now) when you work with TPS/Clarion files.
- Clarion Data Access Layer - an ADO.NET wrapper that translates standard Clarion syntax into ADO.NET calls. This is what most people will use for any database with an ADO.NET provider.
- Direct ADO.NET - like coding in C# but without the curly braces and semicolons.

You can mix and match these techniques. This suggests there won't be a lot of changes in the ABC code since the RTL will know when a database access verb like GET or NEXT is to be handled via interop (Win32 call) and when it's to be handled by ADO.NET.

And having easy, direct access to ADO.NET is hugely cool, especially since it looks like it'll be dead easy to port C# examples to Clarion.NET.

Hangin' with the heavy SQL hitters

Direct link

Posted Tuesday, November 08, 2005 by Dave Harms

If you've been to a Clarion conference, or you read the SoftVelocity news server, you've met or heard of Mike Gorman. You might not, however, realize that Mike is also a high rolling SQL guy, the secretary of the ANSI National Committee on Information Technology Standards (NCITS) H2 Technical Committee on Database. Here's an excerpt from the Committee's most recent annual report:

> H2 met six (6) times since my February 2003 report to INCITS. During that period H2: completed international standardization work on all 9 parts of SQL 2003, the 5th generation of the SQL standard; published the 2nd generation of the full text, spatial, and still image SQL/MM standards; and issued a Technical Corrigendum for the first generation versions the same three SQL/MM standards. In addition, H2 addressed early stage processing of next generation SQL/MM and SQL standards, registering Working Drafts for nine parts of SQL and 5 parts of SQL/MM Framework. The primary focus of new work within H2 was SQL/XML, resulting in the disbanding of the H2.3 Task Group and folding that work into the H2 agenda.

Nice to have Clarion known in such high places!

Clarion Magazine has two articles by Mike:

- Data is Executed Policy: A Case Study (PDF)
- Whitemarsh's Use Of Mimer With Clarion

Mike is also the author of about ten books, at least five of which are available on Amazon.

Better SV web forum links

Direct link

Posted Tuesday, November 08, 2005 by Dave Harms

I've updated some previous blog entries with better links to messages posted in the new SoftVelocity web forums. The problem was that I wanted to link to a specific message in a thread, but at the time I didn't see an obvious way to do that.

For instance, here's a thread called IPD News and disconnected sessions. The URL is http://www.softvelocity.net/community/forums/16/ShowPost.aspx, and that "16" in the URL is the number of the first message in the thread. Each of the replies has its own ID as well - Bob Zaunere has posted a reply in that thread which has a message ID of 23 (I'll explain how I know that in a moment). But if you change the URL to http://www.softvelocity.net/community/forums/23/ShowPost.aspx you'll just go to the start of the thread, not Bob's reply. No help there.

If you look at the page source, however, you'll find that at the start of each message in a thread there's an `a` tag, something like `<a id="_ctl0__ctl1_BodyContentRegion_PostFlatView__ctl0_PostRepeater__ctl1_PostAnchor" name="23">`. The `id` value is unimportant - what matters here is the name value, which is a named anchor, corresponding to the message ID. Add that to the URL and the browser jumps to the anchor location. Perfect! So to navigate directly to message Bob's reply, you simply append the message ID as a bookmark, as in http://www.softvelocity.net/community/forums/16/ShowPost.aspx#23.

Thankfully there's an easier way to find any message's ID than looking at the page source. Just hover your mouse over the message's reply or quote button and look in your browser's status bar for the URL. You'll see something like http://www.softvelocity.net/community/forums/AddPost.aspx?PostID=23, and that number on the end is the message ID.

Someone brighter than me would've noticed this technique being used in the Last Post column of the forum list. Of course, it also took me a few days to realize there was Not Read link...

C7 redirection improvements

Direct link

Posted Wednesday, November 09, 2005 by Dave Harms

Check out this post by SuRF on some of the nifty new redirection features in the C7 IDE:

> For example, in the IDE you will be able to go to the standard File Open dialog and type in the name of a file you want to look at. The system will then look in the directory you are in (according to the dialog), then if the file is not there, it will use the redirection system to find the file.

You can also have nested redirection files, e.g. a rediction file that includes the standard redirection file for defaults. And Viggo has a great suggestion for logging all files used in a project.

Visual Studio Express for Clarion developers

Direct link

Posted Thursday, November 10, 2005 by Dave Harms

You've probably heard by now that Microsoft is giving away free copies of the 2005 Express line of Visual Studio products, including Web Developer, SQL Server, C#, Visual Basic, Visual C++, and Visual J#. There are some important limitations to the Express products, including no MS Office development support (whatever that means), limited web forms, local database connections, no class designer/bench tester, no deployment tools, etc (check the product comparison). The free offer is valid until Nov 6, 2006 - after that time the price goes back up to $49 per product.

If you have the slightest interest in Clarion.NET, then you owe it to yourself to download at least Visual C#. During the install you'll have the option to download the MSDN docs and SQL Server 2005 Express as well, and I recommend you select those options. If you have an earlier version of .NET 2.0 or any of the Express beta products you'll need to uninstall those first.

I recommend Visual C# because that's the language that looks the most like Clarion. It's also the .NET reference language, so if there's something you can do in .NET, chances are you'll find a C# example. And being able to make sure that some C# code works as expected will make porting that code to Clarion a whole lot simpler.

I'll have more to say about using Visual Studio Express in an upcoming issue of Clarion Magazine.

ClarionMag Top 25 (sort of)

Direct link

Posted Friday, November 11, 2005 by Dave Harms

Here's a list of the 25 most popular subscriber-only articles published in Clarion Magazine in the past three years. Actually this list is a little bit biased in favor of older articles, since the longer an article has been up on the web site, the more likely it is to have been read. As well, some articles benefit from being referenced in other ClarionMag articles. Also the spread isn't terrifically wide - the #1 article had just over twice as many views as the #25 article. Even so, #1 and #2 both had about 50% more views than #3, at which point the rating decreases very gradually.

1. Interfaces Everywhere
2. Icons *In* List Box
3. Creating Utilities For MS SQL 2000
4. SQL Identity: Another Approach
5. Clarion 6 First Look: The Examples
6. Integrating The Clarion Report Writer Into Your Applications (Part 1)
7. Calling A Clarion Application With PHP
8. Data Structures and Algorithms Part XX - Topological Sort

9. Product Review: NetTalk, Part 1
10. Data Structures and Algorithms Part XIV - A Queue Is A Queue Is A Queue?
11. Date Filtering with MSSQL
12. Burning COM: How To Write CDs in Windows XP W ith ICDBurn
13. XML For Clarion Developers
14. Making The E-Mail Connection
15. A Calculator Class And Template
16. A String To CString Converter
17. A Class For Tagging
18. Veronica's Short History Of The Windows Operating Systems (Part 1)
19. Clarion, COM, Soap, and HTTP
20. Clarion 6 First Look: The IDE
21. Validating Credit Card Numbers
22. A Tree In A Page Loaded Browse
23. System Tray Popup Windows
24. Browse Greenbars in Clarion 6
25. PostgreSQL 101: Creating Tables And Sequences

FTP Voyager SDK a day late?

Direct link

Posted Monday, November 14, 2005 by Dave Harms

Leroy Schulz posted a note about the FTP Voyager SDK, released on October 31, 2005. I've used FTP Voyager for years, and I'm a big fan of the product. Among other things I use it to securely upload to/download from the ClarionMag server; standard FTP is notorious for its security holes, so if you need FTP, it had better be secure FTP.

If I had to build FTP capability in to a Win32 product, I'd definitely want to take a close look at this SDK, based on my experience with the full product. But how much new development is being done for Win32 these days versus .NET? Of course you can still call Win32 code from .NET, but you lose the benefits of managed code. I have to wonder if Rhinosoft is coming to the developer market just a wee bit too late.

LapLink, Oracle Express free, and save some bux on SQL 2005.

Direct link

Posted Wednesday, November 16, 2005 by Dave Harms

This looks to be a good week for pinching pennies. First up is Rocky Phelps with a tip on getting LapLink Everywhere free for one year. Here's part of the blurb from the LapLink web site:

> Laplink Everywhere delivers powerful remote control with an added web interface that delivers your files, email, PIM data and Internet favorites to ANY web-enabled device-even a PDA or cell-phone!

> The remote control functionality sets you free from the office-Laplink Everywhere also sets you free from being in front of a computer at all. Use your handheld device to find the remote files and emails you need. And with integrated Google Desktop Search, you can find them faster than ever.

The LapLink offer expires November 21, 2005. I assume the software will stop working a year from now unless you pony up the cash, but I didn't see a clarification on the web site.

Adrian Velcich points out that you can freely download Oracle 10/g/ Express Edition for development *and* you can distribute and deploy with no licensing fees. I went far enough through the process to discover the size of the download (157 megs), which required me to go through the license acceptance routine. Please keep in mind that you may *not* use Oracle 10/g/ Express Edition if you are a Specially Designated Terrorist or Specially Designated Narcotic Trafficker or a Denied Person. Think you can use Oracle for WMD development? Hah! Wrong again.

I sure am glad Larry Ellison is protecting the free world.

And I wish the Oracle webmaster didn't feel the need to open a new window on every Oracle link.

Okay, one more item. Paul MacFarlane passes along this tip. With the release of SQL 2005, Microsoft is raising prices by around 10%. You might want to look into buying SQL2000 license with Software Assurance to save some $$.

Clarion IRC chat

Direct link

Posted Friday, November 18, 2005 by Dave Harms

Today's article by Marty Honea on Memory Mapped Files contains a reference to the Clarion chat room on IRC (Internet Relay Chat). You can find instructions on getting connected at Russ Eggen's site. I used to use Pirch98 for IRC, but Russ points out that Pirch98 and Pirch32 have security exploits and are no longer supported by the author. So I've downloaded mIRC. Use the settings Russ supplied and you can get hooked up pretty easily.

On IRC you join a channel, and right now there are two Clarion-related channels, #cw-talk and #cw-source. The former is for discussion, the latter for source postings and other stuff that would otherwise overwhelm chat. If your IRC client has a command line mode (I haven't seen one that doesn't, but I'm a bit out of the loop with the latest IRC clients, as you might have guessed) you join by typing /join #channelname, as in /join #cw-talk. You can probably also pick #cw-talk from a list, although you may have to enter it the first time since it's not exactly a major channel in the world of IRC.

Way back when, IRC chats tended to happen at specific times. Russ tells me that enough Clarion developers are stopping by that #cw-talk is now going 24/7. As I write this, there are 17 people in chat, although at any given time you can expect a certain percentage of those present to be logging, and not actually watching the screen. Some of us have to work, you know. OTOH chat is also one more good reason to run multiple monitors. Put your IRC client off to the side somewhere (mine's running on the right-hand side CRT, while I do most of my work on the LCD in front of me) and you can easily check on the chat while you work.

I also remember lots of problems in the old days with server bandwidth, long ping times, and particularly split chats,

where you'd join the chat, see your name on the list along with everyone else, post a half dozen messages, and wonder why no one replied. Eventually you'd find out that you were on the wrong public server. Everything's working a lot more smoothly now, with a very reliable private server hosted by David Farmer. Thanks, David!

Fat part of the curve still coming

Direct link

Posted Tuesday, November 22, 2005 by Dave Harms

In a recent posting in comp.lang.clarion, Trond Erik Paulsen wrote:

> "If I was chief for a day in SV, I would have dropped all further work on the dot net generator and changed focus to a better IDE for WIN32 apps. That is what we customers want!"

I replied, pointing out the new poll on Clarion Magazine, asking developers which product was more important to them. At 116 responses, 35.3% said C7 was most important, 30.2% favored Clarion.NET.

Sean Hennessy made this reply to me about Trond's comment:

> It would appear by the result so far that he's in the majority if only by a little

I answered:

> Actually to clarify, the poll indicates that C7 is of slightly more interest than Clarion.NET, at this point. If I were to ask how many developers think SV should immediately halt all .NET development in favor of Win32 work, I do think it would be a small percentage, at least among those who have some familiarity with .NET. The analog would be if, just as Clarion for Windows was coming out, someone had taken a poll on how many developers would like to see all Windows development canned in favor of further DOS work. Looking back, that would have been the death of Clarion. And although there's still plenty of life in Win32, .NET really is the future. Clarion.NET is just as important now as Clarion for Windows was then, IMO.
>
> If the poll is at all accurate, then SV's timing looks fairly good. If there were only, say, 10% of developers eager for Clarion.NET, then it might be harder to justify demand. If 70% were eager, it would be easier to argue that SV is late to the party. Certainly some who need .NET now are using other tools (many Clarion developers don't just stick to Clarion anyway - I personally have used Java for server side web stuff for years) but this poll suggests to me that the fat part of the curve is still coming for Clarion.NET.

There has also been some commentary on the "non-scientific" nature of the ClarionMag polls. Trust me, I put these puppies up for entertainment, and I don't take them terribly seriously. I do think that ClarionMag's readership skews the results a little bit in favor of newer technologies, partly because ClarionMag is an online magazine, and online magazines are a much newer technology than print, and partly because my own editorial bias is in favor of newer technologies. .

Dave's personal blog

Direct link

Posted Friday, November 25, 2005 by Dave Harms

I've been bitten by the blogging bug. Too much to say, too few web sites, so I've launched Knobblegrud.com. Come on over and browse around.

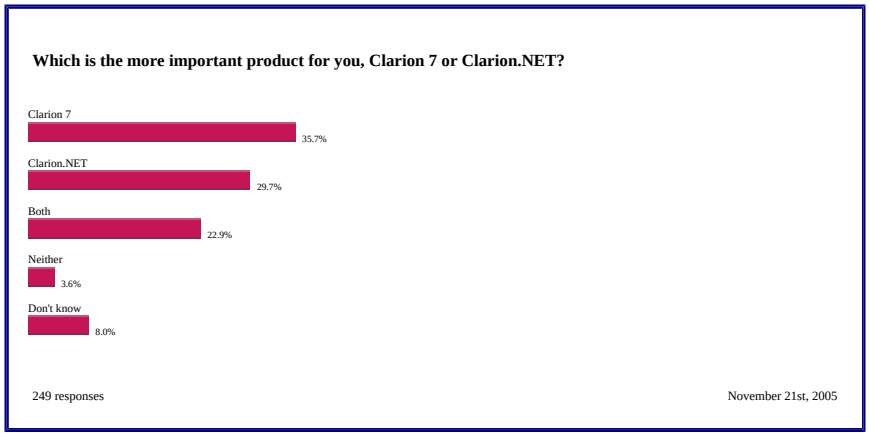
I'll continue to post Clarion-related entries right here, of course.

And it's Clarion 7 by a nose

Direct link

Posted Monday, November 28, 2005 by Dave Harms

A week ago I put up this poll question:



Those are the results as of November 28; for the latest (or final) results see the previous surveys page (you will need a subscription or a free membership to see that page, where you can also view survey results going back to December 2000). For a while there Clarion.NET held a slim lead, but C7 finished strong.

As always, this is an unscientific poll, for entertainment purposes only. I am, of course, entertained by the growing interest in Clarion.NET. But you already knew that, if you've read Part 1 and Part 2 of my "What is .NET and why should I care?" series.

Although SV has indicated there could be a Clarion.NET hand coder's beta before Christmas, I think early 2006 is a more likely target.

CapeSoft Profiler getting good buzz

Direct link

Posted Tuesday, November 29, 2005 by Dave Harms

The fertile minds at CapeSoft have come up with a new programming tool that's starting to get some serious buzz. It's called CapeSoft Profiler, and it'll also set you back some serious cash; the price is $299 in beta, and $399 at release. But users are starting to post some glowing reports in the SoftVelocity newsgroups.

Clarion has had some basic profiling capability for number of years, thanks to a couple of little-known hooks called EnterProc and LeaveProc, which you can override to have your code called when any procedure, routine, or method starts and ends. I gave a presentation on this feature at DevCon 99, and you can find several articles on the subject in Clarion Magazine if you search for EnterProc.

But I don't think this is what CapeSoft's product is using. For one thing, you have to enable the profiler hooks with compiler directives, and CapeSoft's documentation indicates their Profiler only needs Debug Mode set to Full. Also, Profiler is an application that runs your app as a child process.

I should have a copy of Profiler in house for testing shortly - expect a "first look" review in December.

New blog and site lists

Direct link

Posted Wednesday, November 30, 2005 by Dave Harms

I've added two new features to Clarion Magazine:

- A list of Clarion-related blogs
- A list of Clarion-related web sites

If you'd like to be added to either of these lists, send me a link and a short descriptive paragraph.

Both lists are also shown, in part, on the Clarion Magazine home page. Look down the left side. As I expect these lists to grow, I'm only listing the most recently-added items on the home page. On the blog and site pages, however, the entries are sorted alphabetically.

[Last 25 entries] [Last 50 entries] [All entries]