Clarion Magazine

Clarion News

- » RPM Update and Production Schedule
- » SetupBuilder 7.0 Build 2754
- » SetupCast 1.6.0
- » EasyListView 1.01
- » DMC 1.7.0.0 Two Year Anniversary
- » SetupBuilder Ready For Windows 7
- » EasyCOM2INC 2.10
- » 900 Data Management Concepts & Terms
- » Clarion Handy Tools On ClarionLive!
- » ClarionMag's New Free FAQ Site
- » Aussie DevCon ClarionLive Report Time Change
- » If Arnold Can Do It, So Can You!
- » Clarion Third Party Profile Exchange October 1 2009 Release
- » SkinFramework Wrapper Template 1.04
- » SetupBuilder 7.0 Build 2734
- » CFC Library 3.0
- » ClarionLive Aussie DevCon Daily Reports
- » EasyListView 1.00
- » iQ-XML 2.53

[More news]

- » Clarion.NET FAQ
- » Clarion# Language Comparison
- » SV Surprises With A New .NET Template Engine
- » The New Clarion.NET Template Language - Is It Really Microsoft's T4?

[More Clarion & .NET]

[More Clarion 101]

Latest Subscriber Content

All About ClarionMag's New FAQ Site

ClarionMag has a new FAQ site! Who wrote it? Why is it a good thing? Will it remain free? Dave Harms answers the questions.

Posted Friday, October 23, 2009

Source Code Library 2009.09.30 Available

The Clarion Magazine Source Code Library has been updated to include the latest source. Source code subscribers can download the September 2009 update from the My ClarionMag page. If you're on Vista or Windows 7 please run Lindersoft's Clarion detection patch first.

Posted Monday, October 05, 2009

Aussie DevCon Day 1 Notes

The first training day of the Aussie DevCon is in the books, with Bob Foreman providing training Bruce Johnson showing a preview of NetTalk 5, and ClarionLive! providing a video link.

Posted Monday, October 12, 2009

Aussie DevCon Day 2 Notes

During the Day 2 ClarionLive! report David Griffiths, Ted Steward and Bob Foreman talked about Clarion 7.1 and the new DevExpress-based report writer. C7.1 is getting good reviews, and there are suggestions Bob Z may demo an early . NET AppGen.

Posted Tuesday, October 13, 2009

RSSBuilder Again

Steve Parker takes another crack at creating an RSS feed, this time ending up with the sensible solution: Robert Paresi's freely available iQ-XML library.

Posted Wednesday, October 14, 2009

Aussie DevCon Day 3 Notes

Day 3 training had Aussie DevCon attendees pumped about .NET development in Clarion#.

Posted Wednesday, October 14, 2009

ClarionLive!: Code Generation For Clarion.NET

Join ClarionLive! at 9:30pm Pacific Time Thursday October 15th for Robert Zaunere's session - Code Generation for Clarion.Net. This session explains the architecture of the new templates, as well as demonstrating the application generation process for .Net.

Posted Thursday, October 15, 2009

Latest Free Content

- » Source Code Library 2009.09.30 Available
- » SV Surprises With A New .NET Template Engine
- » ClarionMag's New Free FAQ Site

[More free articles]

Clarion Sites

- » Clarion classes, templates and utilities

Clarion Blogs

Aussie DevCon Day 4 Notes

It was a slow news day at the Aussie DevCon as the four day training sessions wound up. Stay tuned for Bob Zaunere's keynote on the .NET templates and AppGen, to be webcast 9:30 p.m. PDT at ClarionLive!

Posted Thursday, October 15, 2009

SV Surprises With A New .NET Template Engine

In his Friday keynote address at the Aussie DevCon, SoftVelocity president Bob Zaunere pulled a rabbit out of the company hat and revealed that the AppGen and template language have been rewritten in .NET, for Clarion.NET. As Dave Harms explains, this has profound implications for the future of Clarion.

Posted Friday, October 16, 2009

ClarionMag's New Free FAQ Site

If you've spent any time on the newsgroups, you know that the same questions are often asked over and over. And every once in a while someone will ask, isn't there a better way? Yes there is. Using the model (and the software) pioneered by the popular programming FAQ site, StackOverflow, Clarion Magazine now brings you the Clarion FAQ. Post your questions; contribute answers; search for information. There's no membership fee to pay, and Clarion FAQ will remain a free access site.

Posted Tuesday, October 20, 2009

The New Clarion.NET Template Language - Is It Really Microsoft's T4?

At the Aussie DevCon, SoftVelocity president Bob Zaunere demonstrated a template written in the new .NET template language. But is it a new template language? Dave Harms argues it's really Microsoft's T4, and explains why that's a good thing.

Posted Thursday, October 22, 2009

Returning Multiple Values

You can call a procedure and not get anything back, or you can call a procedure and get a single value back. But what do you do when you need to return multiple values from a procedure? You pay attention to Dr. Parker.

Posted Monday, October 26, 2009

The Problem With Embeds, Part 2: Extracting And Testing Code

In this second article in the series Dave Harms reworks some of his own embed code into a class, and introduces a new tool for unit testing Clarion code.

Posted Saturday, October 31, 2009

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

Source Code

The ClarionMag Source Code Library

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

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

More info • Subscribe now

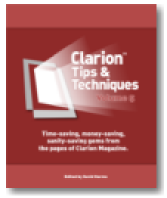
Printed Books & E-Books

E-Books

E-books are another great way to get the information you want from Clarion Magazine. Your time is valuable; with our e-books, you spend less time hunting down the information you need. We're constantly collecting the best Clarion Magazine articles by top developers into themed PDFs, so you'll always have a ready reference for your favorite 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 Tips & Techniques Volume 5 - ISBN 978-0-9784034-1-6
- » Clarion Tips & Techniques Volume 4 - ISBN 978-0-9784034-0-9
- » Clarion Tips & Techniques Volume 3 - ISBN: 0-9689553-9-8
- » Clarion 6 Tips & Techniques Volume 1 - ISBN: 0-9689553-8-X
- » Clarion 5.x Tips and Techniques, Volume 1 - ISBN: 0-9689553-5-5
- » Clarion 5.x Tips and Techniques, Volume 2 - ISBN: 0-9689553-6-3
- » Clarion Databases & SQL - ISBN: 0-9689553-3-9

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

From The Publisher

About Clarion Magazine

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

Subscriptions

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

Satisfaction Guaranteed

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

Dave Harms

ISSN

Clarion Magazine's ISSN

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

About ISSN

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

Clarion Magazine

Clarion News

Search the news archive

Search Engine Profile Exchange November 03 2009 Release

This is a abbreviated online and web updated release - data version.

Posted Wednesday, November 04, 2009

EasyListView Version 1.02 Coming Soon, New Demo.

Changes in EasyListView 1.02 include: File explorer example; export to Excel (TPS example); export to xml (TPS example); Header Word Wrap (TPS example); Hover selection (File explorer example); Hot tracking (File explorer example); Translucent selection (File explorer example); Translucent hot item (File explorer example); Explorer theme (File explorer example); Flag renderer (File explorer example, Attributes column); Item tooltips (File explorer example); Column header font/color (TPS example); Hot item overlay (TPS example); Check boxes (File explorer example). New demo app available. EasyListView is a Clarion wrapper around a .NET ListView.

Posted Wednesday, November 04, 2009

Noyantis Codejock Wrapper Templates Now Codejock v13.2.1 Compatible

New versions of all of the Noyantis Codejock Wrapper Templates are available: 2.01 - CalendarPro Wrapper; v2.02 - CommandBars Wrapper; v1.06 - DockingPane Wrapper; v1.11 - PropertyGrid Wrapper; v1.12 - ReportControl Wrapperl v1.22 - ShortcutBar Wrapper; v2.01 - SkinFramework Wrapper; v2.01 - TaskPanel Wrapper. Modifications include: Codejock v13.2.1 compatibility added. Also, the CommandBars Wrapper tpl has a Bug Fix for a problem that was introduced in v2.01 - your app could potentially GPF in Windows XP. The new version can be downloaded from the Members area using the original download and registration details contained in your sales email.

Posted Wednesday, November 04, 2009

CapeSoft Clarion 7.1 Builds

For those who are brave enough to put them through the test, you can grab the binary installs of CapeSoft accessories for the C71 IDE, Except OfficeInside 3. The non-DLL products have the same install for c70 as c71 so you don't need to re-download those.

Posted Wednesday, November 04, 2009

Xtreme ToolkitPro, Xtreme SuitePro 13.2.1

This is a maintenance and support release that addresses some outstanding issues that were not addressed with 13.2.0, including feature enhancements and fixes. The 40% discount is good towards all ActiveX components and renewal subscriptions through the end of 2009. Codejock SuitePro 2009 vol.3, Codejock CalendarPro 2009, Codejock Command Bars 2009, Codejock Controls 2009, Codejock DockingPane 2009, Codejock PropertyGrid 2009, Codejock ReportControl 2009, Codejock ShortcutBar 2009, Codejock SkinFramework 2009, Codejock TaskPanel 2009. Full license (with 30 Days Support) - XX Developer License with 30 Days Support, ActiveX COM Full license (with 1 Year Support) - XX Developer License with 1 Year Support, ActiveX COM Subscription Renewal (with 1 Year Support) - XX Developer License with 1Year Support, ActiveX COM. To use it select a product, add to cart. After that use Coupon Code (enter it into field in Cart form) CLARION and push Update/Recalculate button. You will see a new discounted price (and small (E) icon)!. After you can Checkout. Pease note that Coupon Code above not suitable for Noyantis wrapper Templates.

Posted Wednesday, November 04, 2009

Aussie DevCon Training DVDs

The upcoming video series from the Aussie DevCon covers the whole seven days of training and conference. There were four audio sources, two CCTV type video sources, one data projector output capture, plus screen capture using Camtasia Studio, plus the original PowerPoint files. These are all being mixed to give the complete professional package. The packages are being broken up into two days of C7 training, two days of Clarion.NET training, and three days of Convention presentations.

Posted Wednesday, November 04, 2009

DockingPane Wrapper Template 1.05

Version 1.05 of the DockingPane Wrapper template is available. Modifications include: Codejock v13.2.0 compatibility added. The new version can be downloaded from the Members area using the original download and registration details contained in your sales email.

Posted Wednesday, November 04, 2009

SkinFramework Wrapper Template 2.00

Version 2.00 of the SkinFramework Wrapper template is available. Modifications include: Codejock v13.2.0 compatibility added. The new version can be downloaded from the Members area using the original download and registration details contained in your sales email.

Posted Wednesday, November 04, 2009

ReportControl Wrapper Template 1.11

Version 1.11 of the ReportControl Wrapper template is available. Modifications include: Codejock v13.2.0 compatibility added; Icons now automatically added to Project List (if filename prefixed with a "~"); Horizontal + Vertical Grid Styles added; Browse reload option added to Print / PrintPreview control templates; New method added - SetGridStyle; New method added - SetRowSelected. The new version can be downloaded from the Members area using the original download and registration details contained in your sales email.

Posted Wednesday, November 04, 2009

TaskPanel Wrapper Template 2.00

Version 2.00 of the TaskPanel Wrapper template is available. Modifications include: Codejock v13.2.0 compatibility added; Save and Restore Layout feature added; Icons now automatically added to Project List (if filename prefixed with a "~"); Font attributes added to Group definition; Font attributes added to Item definition; Markup option added to

Item definition. New Options: Groups Expandable (All Groups); Group Expandable (Individual Group); Hot Track Style; Item Layout (All Groups); Item Layout (Individual Group); Multi Column. Various new class methods have been added including: GetFocusedItem; GetGroupProperty; SetGroupExpandable; GetHotItem; GetItemProperty; SetDefaultItemLayout; SetDefaultMinClientHeight; SetExpandable; SetGroupCaption; SetGroupProperty; SetGroupItemLayout; SetGroupMinClientHeight; SetGroupVisible; SetHotTrackStyle; SetItemAllowDrag; SetItemAllowDrop; SetItemProperty; SetItemCaption; SetItemEnabled; SetItemToolTip; SetItemVisible; SetMultiColumn. The new version can be downloaded from the Members area using the original download and registration details contained in your sales email.

Posted Wednesday, November 04, 2009

PropertyGrid Wrapper Template 1.10

Version 1.10 of the PropertyGrid Wrapper template is available. Modifications include: Codejock v13.2.0 compatibility added; Default Display Text Lines option added to MultiLine Item; New method added - DeleteAll; New method added - DeleteCategory; New method added - DeleteItem; New method added - SetItemDisplayLines. The new version can be downloaded from the Members area using the original download and registration details contained in your sales email.

Posted Wednesday, November 04, 2009

CalendarPro Wrapper Template 2.00

Version 2.00 of the CalendarPro Wrapper template is available. Modifications include: Codejock v13.1.0 and v13.2.0 compatibility added; Manual creation of Schedules simplified; Additional Week / Month view options added; Customizable tooltip formats added; Markup Text facility added to events; Read Only Mode added; DatePicker Class added; Event Handler facility added to both the Calendar and the DatePicker controls; Cell Background Colour facility added; Drag and Drop facility added; Allow Events to be Moved option added; Allow Events to be Resized option added; Display Next / Previous Event Buttons option added; Display Expand Buttons CaptionBar option added; Display Multi Columns CaptionBar option added; Display on a Single Line CaptionBar option added; ImageRsc storage variable increased; Refresh parameter added to SetEventRecurring; Bug fix - CaptionBar would disappear in Timeline Mode; Bug fix - Events could be temporally duplicated if Ctrl key held down while dragging event; Bug fix - Threading Issue in c55 when Language resource used; Bug fix - DatePicker NewDate procedure was not being called when MonthChanged event detected. New procedures added: ClearCustomCellColours_????; SetCustomCellColours_????. Various new class methods have been added including: AddDropTarget; AllowEventMove; AllowEventResize; ApplyOffice2007ThemeFile; MouseClick; GetEventBody; GetEventSubject; GetReadOnlyMode; GetEventRecurringState; GetScheduleCount; ProcessDropTargets; SetReadOnlyMode; SetEventIconAdd; SetEventIconDelete; SetEventLocation; SetEventMarkup; SetEventRecID; SetTooltipFormat; SetMonthCompressWeekends; SetMonthEventEndTimes; SetMonthEventClocks; SetWeekEventEndTimes; SetWeekEventClocks; SetNonWorkingHourColour; SetWorkingHourColour; ShowCaptionBar; ShowCaptionDateLabel; ShowCaptionExpand; ShowCaptionMultiColumn; ShowCaptionScrollDate; ShowCaptionSingleLine; ShowCaptionSwitchView; ShowCaptionTimeLine; ShowEventMarkup; ShowEventNextPrev; Reset. Redundant methods: DefineSchedule; EmptySchedules; SetRemindersObj. The new version can be downloaded from the Members area using the original download and registration details contained in your sales email. PLEASE NOTE, For those users waiting for the ability to remove / modify an individual occurrence of a Recurring Event, then this modification will be available very soon (the code is complete but just needs adding to the TPL).

Posted Wednesday, November 04, 2009

CommandBars Wrapper Template 2.01

Version 2.01 of the CommandBars Wrapper template is available. Modifications include: Codejock v13.2.0 compatibility added; Window Reset added to NewSelection_?? procedure; New method - SetComboSelectionByText; New method - SetCtrlVisible; Bug fix - Initial selection corrected for DropLists; Bug fix - Initial height incorrectly calculated in c6.1; Bug fix - DeleteControl not deleting Ribbon Groups and Ribbon Tabs correctly. The new version can be downloaded from the Members area using the original download and registration details contained in your sales email.

Posted Wednesday, November 04, 2009

ShortcutBar Wrapper Template 1.21

Version 1.21 of the ShortcutBar Wrapper template is now available. Changes include: Codejock v13.2.0 compatibility added; Internal API calls renamed. The new version can be downloaded from the Members area using the original download and registration details contained in your sales email.

Posted Wednesday, November 04, 2009

RPM Update and Production Schedule

RPM for C7.0.5768 will be available for download before end of business (ET) Tuesday, 27-Oct-09. This release includes (for ABC only): Mouse Wheel support in single page and thumbnail view and the fix for a Drill Down bug when a link calls another RPM preview. All changes are documented in source as well as in the Change Log provided in the install. An RPM install for C6.3.9051, source synchronized with the C7 release, will be available by end of business (ET) Thursday, 29-Oct-09. After the AFE install for C7 is released in early November work will proceed on a new AFE server with updated FaxMan libraries. After release, work will resume on RPM/ABC support for Fomin Report Builder and the back port to legacy.

Posted Monday, October 26, 2009

SetupBuilder 7.0 Build 2754

SetupBuilder 7.0 Build 2754 is now available. This release is available, free of charge, to all SetupBuilder customers who have an active SetupBuilder maintenance subscription plan. The update contains some important bug fixes and improvements.

Posted Monday, October 26, 2009

SetupCast 1.6.0

Version 1.6.0 of SetupCast is available for download. This is a free upgrade to all registered users. Just use the Check for Update feature or download a fresh copy. This release provides Windows 7 compatible installers and full support for automatic publishing of SetupBuilder 7 projects.

Posted Monday, October 26, 2009

EasyListView 1.01

EasyListView 1.01 is now available. Changes include: EasyListView.dll assembly is signed now so you can add EasyListView.dll into the GAC; There is a new "Link oleautcg.lib" setting added in the Global template; ListView Previewer; Possibility to set a Font for the whole ListView (Appearance > General); Possibility to set a Font for the EmptyList message (Appearance > General); Hot Items (items under the mouse cursor); Possibility to set a Font, Styles

and Colors for the Hot Item (Appearance > General); Example of using Hot Item in "TPS example" (added also a new "Use Hot item" check box); New methods: GetUseHotItem, SetUseHotItem and SetHotItemStyle. A new demo is available.
Posted Monday, October 26, 2009

DMC 1.7.0.0 Two Year Anniversary

DMC 1.7.0.0 is now available. This release contains many new features. If your Maintenance Plan has expired you will need to renew your licence and apply the new one before updating, otherwise you will not be able to update. The price of DMC has increased from 149 Euros to 199 euros, and there is also a small increase in renewal pricing.
Posted Monday, October 26, 2009

SetupBuilder Ready For Windows 7

SetupBuilder creates install programs for Windows 7 and Windows Server 7 aka Windows Server 2008 R2. SetupBuilder 7 Professional Edition starts at $169.00 and the Developer Edition starts at $299.00 for a royalty-free usage license. A trial version is available.
Posted Monday, October 26, 2009

EasyCOM2INC 2.10

Posted Monday, October 26, 2009

900 Data Management Concepts & Terms

Mike Gorman has completed a 250 page book that contains over 900 Concepts and Terms related to Data Management.
Posted Monday, October 26, 2009

Clarion Handy Tools On ClarionLive!

On this Friday's ClarionLive! (October 23, 2009) Gus Creces will present a survey of Clarion Handy Tools (CHT) and dive into a few of the popular areas, including Application Configuration, Coding Assistance, Browses, Dates and Calendars, FTP, HTTP and more. There are over 70 of these "broad" categories and more than 380 templates in the CHT toolkit.
Posted Friday, October 23, 2009

ClarionMag's New Free FAQ Site

If you've spent any time on the newsgroups, you know that the same questions are often asked over and over. And every once in a while someone will ask, isn't there a better way? Yes there is. Using the model (and the software) pioneered by the popular programming FAQ site, StackOverflow, Clarion Magazine now brings you the Clarion FAQ. Post your questions; contribute answers; search for information. There's no membership fee to pay, and Clarion FAQ will remain a free access site.
Posted Tuesday, October 20, 2009

Aussie DevCon ClarionLive Report Time Change

The scheduled time for the Aussie DevCon report at ClarionLive has been moved up to 6:30 p.m. Pacific time.
Posted Monday, October 12, 2009

If Arnold Can Do It, So Can You!

This week's ClarionLive webinar features Arnold Young and John Hickey, as Arnold builds an MSSQL app using everything he's learned in the past few months. Tasks include: Loading MS-SQL Express 2005 Server; Creating a SQL database from CVS, XLS, and TPS files; Building the associated Clarion data dictionary and application; Utilizing PROP:SQL via a donated class and via a third party template product; Creating the deployment of the application including installing a local MS-SQL Express Server and populating initial data into the new SQL database; Creating a separate application for data structure maintenance via FM3. Take the surveys at www.clarionlive.com, and enter to win a Clarion Live DVD. And starting this Sunday through Friday at 7pm PDT, watch a live webcast from Aussie Devcon. Visit www.clarionlive.com for details. Please note the time is now 7pm (not 8pm as previously announced), because daylight savings time started in Australia last weekend.

Posted Thursday, October 08, 2009

Clarion Third Party Profile Exchange October 1 2009 Release

The Clarion Third Party Profile Exchange has been updated. This is a web updated release - data version.

Posted Monday, October 05, 2009

SkinFramework Wrapper Template 1.04

Version 1.04 of the SkinFramework Wrapper template is now available. The new version has been uploaded to members area along with a new demo example. Changes include: Exclude Modules options now specified prior to the LoadSkin method call; New sample skins added to demo application. The new version can be downloaded from the Members area using the original download and registration details contained in your sales email.

Posted Monday, October 05, 2009

SetupBuilder 7.0 Build 2734

Lindersoft has released SetupBuilder 7.0 Build 2734. This release is available, free of charge, to all SetupBuilder customers who have an active SetupBuilder maintenance subscription plan. The update contains some important bug fixes and improvements.

Posted Monday, October 05, 2009

CFC Library 3.0

CFC Library 3.0 is now available. The library consists of a set of classes and templates for creating tooltips, processing the hot keys and for creating the WinAPI menus and toolbars.

Posted Monday, October 05, 2009

Clarion Magazine

SV Surprises With A New .NET Template Engine

by Dave Harms

Published 2009-10-16

Note: The following comments are based on the ClarionLive! webcast of Bob Zaunere's Aussie DevCon presentation. The video and audio link imposes certain limitations; I apologize in advance for any inaccuracies. And my sincere thanks to Geoff Spillane, Arnold Young and John Hickey for making the webcast happen.

In his opening address to the Aussie DevCon, SoftVelocity president Bob Zaunere gave a brief but remarkable presentation on the new Clarion.NET Application Generator.

In short, the new AppGen is a game changer.

Bob described how, in the process of developing the new Clarion.NET AppGen, they went down the path of creating .NET templates using the existing template language. And they found out that although they have a powerful Win32 generator, one thing made that product possible: Clarion's own special set of user interface controls (buttons, list boxes, etc).

In .NET, all that changes. The standard set of controls is much larger, and beyond that is a whole world of third party controls.

The short path was to support the standard controls only, using the existing template engine.

The longer path, the one SV is taking, is a new AppGen built on a template language that is 95% written in C#. These templates are compiled into assemblies, so for the first time Clarion will have compiled templates. And because the templates are .NET code you can debug them with the .NET debugger.

Bob Z pointed out that in any complex system the vast majority of the code is not the user interface. The AppGen he demonstrated wasn't really an AppGen, more a temporary interface (created over the course of a week) on top of the new template engine. On the left side pane was a procedure list. On the right, a set of property grids. It was difficult to see on the web cast, but the point wasn't so much the crude demo interface but the way template prompts were presented in property grids. Each property in a template can have its own property editor, which is a far more configurable system than the current AppGen.

Design goals for the new AppGen include:

- Version control - the app format is 100% XML and version control friendly (Subversion)
- You have access to any .NET control/component/service from the template language
- The ability to attach developer documentation/notes to app and or to individual procedures.
- Option to generate test cases for aspects of the APP (nUnit)
- Support fo all Clarion-standard code templates, e.g. call procedures on lookup, instantiate procedure on thread, export to XML
- Template support for Actions for any .net control including controls from third party vendors ("best efforts")
- Template language code used in the first release is C# (for ease of debugging), subsequent releases can use any .NET language.
- Layered interface - you can determine the level of detail you want to work with
- A much flatter interface

Design goals for the generated code include:

- All of the business rules and business validation logic will be generated in a domain model tier and not within the UI tier
- The data access tier uses LINQ via SV's LINQToFileProvider, LINQToSQL, and future LINQ providers
- The data access tier can use the managed .NET IP driver
- The LINQToFileProvider incorporates an ABC-style framework - think of it as the best of ABC on top of LINQ
- LINQ to SQL is itself a lightweight ORM; the templates are being designed so that in the future they could generate using other heavier ORMs such as Entity Framework, NHIbernate, CSLA etc.

The intent is to be able to create code for WinForms, WebForms, CompactForms and web services. Apps generated for each target platform use essentially the same design for the generated code.

Bob asked for requests and ideas for the new AppGen. It was difficult to hear some of the questions over the web link but there was some discussion around dictionaries, including support for multiple dictionaries.

Closing quotes from Bob Z's presentation:

> Imagine this - you will generate Clarion styled database applications for every platform supported by .NET without having to be a .NET expert or even writing any .NET code
>
> Each target platform utilizes essentially same design pattern for the generated code (learn one and know them all)

Conclusions and impressions

I was pleasantly surprised (actually gobsmacked is probably a better term) by Bob Z's announcement of a new .NET template engine written in C#.

It may be difficult for most Clarion developers to fully appreciate the significance of this project. One very good way is to spend a whole lot of time writing a template chain in C6 or C7 to generate .NET code, as I've been doing over the last year and a half. You quickly come to appreciate the unique power of the template language, as well as the maddening frustration of not being able to extend the template language itself with .NET code for all those tasks for which the template language, as it is now, is poorly suited. And you wish often for a real template debugger, but you settle for OutputDebugString and DebugView.

In my case I'm writing templates for MVC web apps (in support of, among other things, a new ClarionMag site) and thanks to the nature of the MVC pattern I don't need to rely on the C7 window designer for much more than simply placing controls on a form in a given sequence. Position, appearance is all up to the style sheets. But my, it would be so nice to be able to invoke a different editor here, or a certain bit of .NET code there.

All of that is what the new template engine offers.

I've long argued that the great value of Clarion is not in the Clarion language itself. Oh, the language has its fans, and probably a majority of Clarion developers think much more highly of it than I do.

But let's face facts: although we're a (sort of) loyal and (sometimes) tightly-knit community, we don't have a lot of young blood. If Clarion is to survive, and thrive, it has to grow beyond its current borders. And it will do that by becoming a compelling solution for folks who aren't interested in the Clarion (or Clarion#) language.

The new template engine offers that hope. Until now it's been a daunting task to create an entire application template chain for other languages, in large part due to the limitations of the AppGen and the near impossibility of debugging the templates.

It appears likely that SoftVelocity will focus on Clarion# templates, and that's as it should be. Let others bring out the C# and VB.NET templates, or perhaps IronRuby templates - who knows?

The new template engine has the potential to liberate Clarion's code generation technology from its Clarion-centric limitations. And that would be a very good thing, both for the Clarion community and for the wider world of .NET programming. Even if you don't care one iota for .NET, if you rely on Clarion in any form you surely care that SV survives and prospers.

But a note of caution is also in order. Keep in mind that:

- This is an early demo - it may be some time before a usable version appears.
- SV has its existing Win32 customer base to tend, and that takes resources too. A solid 7.1 release is essential.
- The plan for a common architecture for WinForms, WebForms and CompactForms is ambitious.

In other words, the proof is in the pudding. But certainly the direction SV is taking looks most promising, and much better than I had hoped. I believe a fully .NET template language and AppGen is exactly the right approach.

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

Reader Comments

Posted on Friday, October 16, 2009 by Steven Sitas

Hello David,
well this should have been the "direction from the beginning".
The problem now is that this really is a Huge project and it will probably take sometime - probably over a year from now ..

Good news and bad news at the same time :)

Posted on Friday, October 16, 2009 by Dave Harms

I agree - it's a huge project, hence my cautions. But I'm very glad to see this direction.

Dave

Posted on Friday, October 16, 2009 by Dave Harms

A clarification about the compiled templates. I believe Z's point was that this adds to the flexibility - it doesn't take anything at all away. Vendors can (and for all we know may have to) still provide the templates as source.

In C7 when you register a template it's stored up in the registry.trf file along with the other templates. No software in the world other than the Clarion AppGen knows what to do with a TRF file.

In the new AppGen, when you register a template (as I understand it) the IDE compiles the template into a DLL. a .NET assembly. Among other things, that means you can debug that template DLL.

It also suggests that you could ship templates as DLLs if you wanted to protect your intellectual property. Whether the developer community will accept that is another point - I suspect that, if possible, it won't be common.

Dave

Posted on Saturday, October 17, 2009 by Arnor Baldvinsson

I think compiled templates are a good idea and I would also like to see abilities to use more code, like C++/C#/Whatever in the templates to make them more powerful. However, it sounds like we are still a year away from a gold releaes of Clarion.net and then another year until the kinks are ironed out - 2011-2012 for Clarion.net probably. Can we wait that long?

Best regards,

Posted on Saturday, October 17, 2009 by Dave Harms

One very important point about Clarion.NET vs C7 is that (assuming 7.1 is as solid as it seems) the rest of the IDE is already usable.

The template engine, I think, should be solid pretty soon, since by its nature it's highly testable.

The dictionary interface to the template engine seems pretty straightforward, and the dictionary editor is basically done.

It'll be interesting to see how they implement the window designer into the new AppGen. I'm guessing that's going to be the biggest task by far.

Dave

Posted on Monday, October 19, 2009 by Majodi Ploegmakers

I don't think LINQ to SQL is a good choice. I believe it is discouraged by MS and it only works (again I think) with MSSQL. SQL to Entities would be a better choice or (I'm not an expert) solutions like NHibernate.

Also, wouldn't it be better to pick WPF instead of WinForms?

Posted on Monday, October 19, 2009 by Dave Harms

Majodi, I have the same concerns about LINQ to SQL, although Z did point out that SV's new LINQtoFileProvider adds some ABC-like code to help manage relationships etc.

By creating their own LINQ provider they have the opportunity to add support for TPS (although I don't know if that's been explicitly stated - perhaps someone can clarify) as well as keep the code a bit simpler. I've worked with NHibernate, and although I like it a lot and am using it for the new ClarionMag site, it does add a bit more complexity in terms of the classes you need to generate (entities, mapping classes etc.).

I do think that down the road there absolutely will have to be C# templates that employ NHibernate and/or EF. NH is more mature, not only because it's been around longer but because it's a port of the long-standing Java project. EF 4 (which is really version 2) looks to be much more usable than EF 1, but it's still in beta and doesn't have support for as many SQL databases as NH. Also if you're not on MS SQL you have to worry a bit about being a second class citizen in EF. But it'll be a strong contender I've no doubt.

As for WPF, I think SV is largely relying on the SD folks. There was a WPF designer in SD 3.1, but it was withdrawn after Microsoft announced they would be adding a xaml parser to .NET 4.0. A WPF designer is in the works for SD 4.0, and at some point presumably SV would be in a position to look at implementing that for the new AppGen.

Dave

Add a comment

Clarion Magazine

The New Clarion.NET Template Language - Is It Really Microsoft's T4?

by Dave Harms

Published 2009-10-22

At the Oz DevCon Bob Zaunere announced a new .NET template language and AppGen. To begin with, for the initial template sets the template language itself is C#, which is significant in that it makes the entire .NET framework available to template writers.

Bob Z indicated that the template *engine* was largely complete, but the UI (the part of the AppGen we interact with) had not yet been done, other than the very basic UI code cobbled together for the demonstration.

But what's all this about a template engine? How is that different from the AppGen?

As I understand it, in Clarion.NET the template engine and the AppGen user interface are two different units of code (perhaps two different DLLs) that together make up what we think of as the AppGen.

Keep in mind that all of what I'm saying rests on a bunch of (hopefully well-informed) assumptions on my part.

There are a couple of different jobs the AppGen has to do. One is to present a user interface, which in the Win32 product is a window showing various views of the procedure tree, from which you can bring up other windows that provide further information on the application. Some of those windows and prompts are hard coded (there's *always* a procedure window with certain prompts, for instance), and others are created on the fly based on what's in the templates.

Besides presenting windows and prompts to the programmer, the AppGen also has to collect programmer responses and store that information in the APP file. All of that activity makes up the UI layer, if you like.

The other layer is the code that knows how to merge the templates and the APP information and generate source code. My understanding is that when Bob Z talked about the template engine, he meant this layer.

An example

But what does the new template language look like? Bob Z showed a template on screen, but it was difficult to make out the text over the ClarionLive! webcam. Fortunately, Diego posted a snippet which I suspect is the same template Bob Z showed (line breaks added):

```
<#@ template language="C#" debug="true" type="Utility" ↵
  name = "DCTTree" family = "Clarion"#>
<#@ assembly name="mscorlib.dll" #>
<#@ assembly name="System.dll" #>
<#@ assembly name="System.Drawing.dll" #>
<#@ assembly name="System.Design.dll" #>
<#@ assembly name="System.Drawing.Design.dll" #>
<#@ import namespace="System.Collections.Generic" #>
<#@ import namespace="System" #>
<#@ import namespace="SoftVelocity.TextTemplating" #>
<#@ import namespace="System.Drawing" #>
<#@ import namespace="System.Drawing.Design" #>
<#@ import namespace="SoftVelocity.DataDictionary.Design" #>
<#@ property processor="PropertyProcessor" name="OutputFile" ↵
  type="System.String" defaultValue="none" ↵
  editor="System.Windows.Forms.Design.FileNameEditor"#>
<#+ string _OutputFile = "";#>
<# this._OutputFile = this.OutputFile;#>
<# Open(this.OutputFile);#>
//Print out for DCT:<#= this.DataDictionary.FileName #>
<#    foreach (ITable t in this.DataDictionary.Tables)
   {
    foreach (IKey k in t.Keys)
    {#>
//AttributeAutoNum= <#= k.AttributeAutoNum.ToString()#>
//AttributeCase= <#= k.AttributeCase.ToString()#>
//AttributeExclude= <#= k.AttributeExclude.ToString()#>
//AttributePrimary= <#= k.AttributePrimary.ToString()#>
//AttributeUnique= <#= k.AttributeUnique.ToString()#>
//DoNotAutoPopulate= <#= k.DoNotAutoPopulate.ToString()#>
//ExternalName= <#= k.ExternalName#>
//KeyType= <#= k.KeyType.ToString()#>
//Table= <#= k.Table.Name#>
//IsInRelation = <#= k.IsInRelation()#>
<#     foreach (IKeyComponent kc in k.Components)
     {#>
      //Col= <#= kc.Column.Name#>
      //IsAscending= <#= kc.IsAscending#>
     <#}
    }#>
    //Columns:
    <#foreach (IColumn c in t.Columns)
    {#>
     //Name= <#=c.Name#>
    <#}
    foreach (IRelation r in t.Relations)
    {#>
```

```
      //IsOneToMany= <#=r.IsOneToMany(t)#>
      //Related= <#=r.Relation.Name#>
     <#}
    }#>
<# Close();#>
```

Is it T4?

It appears that SoftVelocity's new template language is, in fact, Microsoft's T4 template language. I've been hearing about T4, I've even looked at some code. But I didn't put the two together until I read Dennis Evans' comment on ClarionFAQ about the new template language looking a lot like T4.

SV adopting T4 isn't a bad thing; in fact, it's a very good thing. As with numerous other aspects of the product, SoftVelocity appears to have moved beyond the not-invented-here phase and is quite willing to use off-the-shelf goods to enhance its own product.

The thing about T4 is it's not a complete templating *system*. It's just a language, a standard way of doing code generation. Critically, a T4 template really isn't set up to interact with other T4 templates. It more or less exists in isolation. In Clarion we have template *chains*, which is really a fancy way of saying we have a bunch of different templates that interact with each other, usually in the process of creating an entire application.

If you want T4 templates to interact with each other the way Clarion templates interact with each other, you need to come up with a *template harness*. That's a wee bit of code (okay, a bucketload) that can handle more than one template at a time, that can make all the different templates do their respective jobs so you can create your application.

So no, in fact, SV hasn't created a new template language.

Among other things, that's good because there are existing tools that can help template developers, such as the Clarius (yes, that's their name) T4 template editor.

How the template works

So let's have a look at the sample template, in the light of T4.

The <# and <#@ strings are delimiters to allow the template engine to differentiate between the template source and the generated source. If you're going to include .NET in template statements you need a way to separate the two. So why choose the template language? Why not delimit the generated source? Simply because while the template statements never include generated source, but the generated source may include template statements.

This line is equivalent to a #Utility statement:

```
<#@ template language="C#" debug="true" type="Utility" ↵
  name = "DCTTree" family = "Clarion"#>
```

Remember that this template is going to be compiled into a temporary class. And if you're going to compile a .NET application you'll almost certainly need to reference some other assemblies. The assembly directives tell T4 which DLLs to add as references to the temporary class project so the class will compile.

```
<#@ assembly name="mscorlib.dll" #>
<#@ assembly name="System.dll" #>
<#@ assembly name="System.Drawing.dll" #>
<#@ assembly name="System.Design.dll" #>
<#@ assembly name="System.Drawing.Design.dll" #>
```

The import directives are equivalent to C# using statements:

```
<#@ import namespace="System.Collections.Generic" #>
<#@ import namespace="System" #>
<#@ import namespace="SoftVelocity.TextTemplating" #>
<#@ import namespace="System.Drawing" #>
<#@ import namespace="System.Drawing.Design" #>
<#@ import namespace="SoftVelocity.DataDictionary.Design" #>
```

Property processors are property editors. The top level view in the AppGen, at least in the rudimentary UI shown in Eden, is a list of what we think of procedures (like the procedure tree), and for each "procedure" you have one or (presumably) more property grids. Each item in the grid, which you can think of as being a template prompt, can have one of the standard property editing behaviors (true/false flag, file dialog, pick from list, etc.) or it can have a more specialized property editor.

The following code defines a property processor named OutputFile which uses the standard System.Windows.Forms.Design.FileNameEditor class, which presents a file dialog. FileNameEditor is derived from UITypeEditor, so presumably you can create your own custom property processors by deriving from that class.

```
<#@ property processor="PropertyProcessor" name="OutputFile" ↵
  type="System.String" defaultValue="none" ↵
  editor="System.Windows.Forms.Design.FileNameEditor"#>
```

Next the template declares a string variable. The template is actually being compiled into a single method, so any declaration in the code, as with the _OutputFile string, will result in an inline declaration in the method. The <#+ syntax tells T4 to create the string as a class variable; you would use the same syntax if you wanted to create a new method in the temporary class.

The template assigns the string to the value returned by the open file dialog, and opens the file for output.

```
<#+ string _OutputFile = "";#>
<# this._OutputFile = this.OutputFile;#>
<# Open(this.OutputFile);#>
```

I don't know that the comment // characters at the beginning of this line (or the other comment lines) are significant. In any case you can see that this is generated code which contains a bit of template code:

```
//Print out for DCT:<#= this.DataDictionary.FileName #>
```

It looks to me like the kinds of things I've done in templates where I output comments in generated Clarion source like this:

```
! File: %File
```

Now it's time to loop through the tables and keys. This is equivalent to #For(%File) in the Clarion template language, followed by #For(%Key).

```
<#    foreach (ITable t in this.DataDictionary.Tables)
   {
    foreach (IKey k in t.Keys)
    {#>
```

In the Clarion language, %File and the other built-in symbols are read-only, and their values are loaded by some magic code. It looks like these symbols will be accessible in Clarion#. I passed along a question to that effect, via Stu, and Bob Z seemed to indicate that yes, you could substitute your own data for the dictionary data. That might seem like an unimportant thing, but it opens the door to bypassing the dictionary entirely; I know of some developers who would prefer to get their metadata directly from the SQL database, rather than have to manage that data in the dictionary editor. And I'm interested because I'm using the NHibernate ORM in my .NET coding and I want to work with a massaged version of the table structure, one that corresponds to a NHibernate object model. In that scenario I might in fact have a dictionary, but use the dictionary to build an application that would generate an object model of my database so I could use the object model in my APP. So many possibilities!

Again, the following appear to simply be comments. The ToString() method probably should not be needed.

```
//AttributeAutoNum= <#= k.AttributeAutoNum.ToString()#>
//AttributeCase= <#= k.AttributeCase.ToString()#>
//AttributeExclude= <#= k.AttributeExclude.ToString()#>
//AttributePrimary= <#= k.AttributePrimary.ToString()#>
//AttributeUnique= <#= k.AttributeUnique.ToString()#>
//DoNotAutoPopulate= <#= k.DoNotAutoPopulate.ToString()#>
//ExternalName= <#= k.ExternalName#>
//KeyType= <#= k.KeyType.ToString()#>
//Table= <#= k.Table.Name#>
//IsInRelation = <#= k.IsInRelation()#>
```

For each key, loop through the key fields and print out the name and the ascending/descending flag:

```
<#     foreach (IKeyComponent kc in k.Components)
     {#>
      //Col= <#= kc.Column.Name#>
      //IsAscending= <#= kc.IsAscending#>
     <#}
    }#>
```

List the column names:

```
    //Columns:
    <#foreach (IColumn c in t.Columns)
    {#>
     //Name= <#=c.Name#>
    <#}
```

List the related tables and the relation type:

```
    foreach (IRelation r in t.Relations)
    {#>
     //IsOneToMany= <#=r.IsOneToMany(t)#>
     //Related= <#=r.Relation.Name#>
    <#}
```

Finish and close the output file:

```
    }#>
  <# Close();#>
```

Aside from the dictionary data, this is pretty much a bog standard T4 template.

The real magic, which we haven't seen yet, is how the Clarion.NET template engine/harness will stitch together the many different templates and types of templates required to create a full-blown application. That's also the bit that (as far as I know) is still missing from the kinds of work being done with T4.

Summary

The key benefits of the new template language include:

- Ability to write and debug in C# (and eventually other .NET languages)
- Full use of the .NET framework
- Control over the dictionary data (hopefully)
- The ability to write custom property processors

The "new" template language is almost certainly Microsoft's T4. That both gives some instant credibility to the new AppGen and removes a large burden of responsibility for making sure the underlying templating system works property. The task ahead for SoftVelocity is to create an AppGen that brings to T4 templating the same power (and more) that Clarion developers have enjoyed in the Win32 world.

Resources

- To better understand T4 I suggest you start by reading Kathleen Dollard's article on T4 in the May issue of Visual Studio Magazine. Kathleen is a long-time champion of code generation in .NET.
- MSDN has a fair bit of documentation, of course.

- Oleg Sych's blog has some great information. (See if you can find the smoking gun in the SV template example.)

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

Reader Comments

Posted on Friday, October 23, 2009 by Lee White

What would REALLY make it cleaner, and easier to read, is if you could delineate sections instead of every line, as you can in ASP...

<#
yada
yada

yada

yada
yada
#>

That would make it FAR less cluttered and also require less MASHING of the keyboard!<g>

Posted on Friday, October 23, 2009 by Dave Harms

You can do that, Lee. Check out the first ForEach loop.

Dave

Posted on Saturday, October 24, 2009 by Kevin Dohren

Hi David,

First the good news.

I've just downloaded the ISO of MS Express 2010 Beta 2
and T4 text transformations work within the IDE, although the command line tool TextTransform.exe seems to be missing.

(T4 wasn't previously available in the Express versions)

So we can all have a 'play' with T4 for free.

Now the slight rant - why didn't SoftVelocity say straight out that the new template language was based on T4 .

Just as in the case of DevExpress and the new ReportWriter this has had to be 'discovered' by the community.

There's nothing wrong with using either DevExpress or T4 (or any other external component for that matter) - but to be less than open about it is counter productive.

Posted on Sunday, October 25, 2009 by Luuk Sluijter

Thank you Dave, clear description about T4. Will we have in the future two different template languages, one for Clarion 7 and one for Clarion.NET, that both are used form the same IDE, or will it merge into ONE language that mixes the old and the new template format?

Posted on Sunday, October 25, 2009 by Rex Kersley

If I remember correctly Bob Z. spoke about xml for the .NET templates, not T4. Now, I don't know anything about T4, maybe it is xml based, otherwise how did T4 get into the loop?

Posted on Monday, October 26, 2009 by Dave Harms

Kevin,

Cool, thanks for pointing that out. If you want to use an T4 editor like Clarius (which is a plugin) you'll need a paid version of VS, however.

Dave

Posted on Monday, October 26, 2009 by Dave Harms

Luuk,

I don't know, but it seems to me that backporting the Clarion.NET template language to Win32 would be a lot of work, for perhaps not that much benefit.

Dave

Posted on Monday, October 26, 2009 by Dave Harms

Rex,

No, I'm not aware either of Z actually saying anything about T4 at the conference.

Diego (who works for SV) posted a template (see above) which I believe is the same one Z showed.

The syntax of that template is identical to T4 syntax.

T4 does have an XML-ish syntax.

But as I noted in the article, T4 alone isn't a replacement for the Clarion template language; it's more like a good foundation on which SV can build.

Dave

Posted on Wednesday, October 28, 2009 by Dave Harms

Here's another T4 editor:
http://t4-editor.tangible-engineering.com/T4-Editor-Visual-T4-Editing.html

Dave

Add a comment

Clarion Magazine

All About ClarionMag's New FAQ Site

by Dave Harms

Published 2009-10-23

As you may have already heard, ClarionMag has launched a new Clarion FAQ (frequently asked questions) site at faq.clarionmag.com. And as much as we'd like to take credit for the site itself, we can't. You might know that a while back the good folks over at Fog Creek (makers of FogBugz) came up with a FAQ site for programmers of all stripes. That site is called StackOverflow, and it's been a huge success. In fact, it's been such a success that lots of people have asked Fog Creek if they can have their own StackOverflow for their own communities, many of which are not even programming-related.

Fog Creek responded to this demand by creating StackExchange, a service that provides hosted StackOverflow-like sites. And that's what ClarionFAQ is - a hosted site, paid for and administered by Clarion Magazine and tailored to Clarion programming, but otherwise created and run by Fog Creek.

Why another FAQ system for Clarion?

Yes, there have been other FAQ systems for Clarion. Why is this one better? Because it's built on the proven model of StackOverflow, and it has the support of Clarion Magazine. We believe that in time, just as Clarion Magazine has become the web's number one resource for quality Clarion programming information, so will ClarionFAQ become the number one resource for Clarion questions and answers.

Will ClarionFAQ always be free?

ClarionFAQ will remain a free site. Yes, there are costs, which are being paid by Clarion Magazine. We intend to recoup those costs, of course, so you'll forgive us for splashing a few ClarionMag ads about the place. And perhaps other ads, in time.

How can I keep tabs on what's happening at ClarionFAQ?

If you want to be automatically notified whenever new questions are posted to the FAQ, get an RSS reader (many email and news readers offer this functionality) and add the ClarionFAQ recent questions RSS feed.

At some future point we'll also add the RSS feed to the ClarionMag home page, but that probably won't happen until after we go live with our new site.

Does this diminish Clarion Magazine in any way?

Not in our opinion. If anything, we see the relationship between ClarionFAQ and ClarionMag as mutually beneficial. ClarionMag's role has always been to provide detailed, well-written, peer reviewed, professionally edited material. That need will always be there, and ClarionFAQ will provide important feedback about the kinds of issues that are important to Clarion developers.

Is this a part of the "new site" you keep talking about?

Although ClarionFAQ is an important new development, it's not actually part of the major upgrade coming to the Clarion Magazine site. You'll have to wait a little longer....

If you have any other questions about ClarionFAQ, please post them... at ClarionFAQ!

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

Reader Comments

Add a comment

Clarion Magazine

Aussie DevCon Day 1 Notes

by Dave Harms

Published 2009-10-12

The 2009 Aussie DevCon is now underway in Eden, Australia. The folks at ClarionLive! (Arnold Young, John Hickey, and John Hamilton) and DevCon organizer Geoff Spillane have set up a live feed from the conference. There are daily reports at 7 pm Pacific time, and these reports are being recorded so you can watch them at your convenience. There may also be some live feeds which are not recorded - if you were up late Sunday night (North American time) you could have watched Bob Foreman conducting training. At the time I signed off the camera view was a fairly wide shot of the front of the room, and the projection screen wasn't at all readable, but the audio was clear.

A few notes from the first day:

- Bob Foreman didn't have an official date on C7.1 release, saying that "hopefully it won't be much longer."
- Bob Z arrives on Wednesday, but probably isn't expected at the venue until Thursday.
- Bob F said he didn't know what Bob would be bringing with him for the .NET presentation. If there are no visuals [of the code generation process], then there would be as much possible information about the process.
- There are plans to have templates for WinForms, CompactForms, and ASP.NET.
- Bruce Johnson was asked about 7.1 (which Bob Foreman is using during the training session) and offered the opinion that it's looking pretty good. Lots of polish.

You can also see Bruce Johnson's NetTalk 5 preview presentation at ClarionLive! Key features of this new release include:

- JQuery integration - this is now a very important part of NetTalk 5
- Multi-site host server, so you can have multiple non-SSL sites running on one IP address; you can also update a site without losing the session data
- PHP support (requires OddJob)

There are a number of Javascript UI libraries out there, but JQuery is looking more and more like the frontrunner. Both Microsoft and Nokia have adopted JQuery for use in their development platforms. One of JQuery's key features is its lean core coupled with a plugin architecture. This makes it easier to keep your Javascript downloads as small as possible while still allowing you to add individual new features. The plugin design is a big part of JQuery's success.

NetTalk 4 is already a popular choice for developers looking to create web sites in C6; NetTalk 5 looks like it will make the choice of Clarion Win32 web development tools a no-brainer.

NetTalk 5 integrates ThemeRoller support, which implements flexible UI theming.

If you purchased or upgraded to NetTalk 4 after January 1, 2009 then you get NetTalk 5 free. Otherwise the upgrade price, for a limited time, is $299, regular $399. The new license is $499 for a limited time, regular $599.

Finally, here's a screen cap of the ClarionLive chat, showing (from left to right) John Hickey, Bob Foreman, Pierre Tremblay and Arnold Young.

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

Reader Comments

Add a comment

Clarion Magazine

Aussie DevCon Day 2 Notes

by Dave Harms

Published 2009-10-13

On the second day of training attendees at the Oz got some more exposure to Clarion 7.1 and the new report writer. Conference organizer Geoff Spillane reports that attendees are very happy with what they're seeing of 7.1 in Bob Foreman's sessions - features and stability are great. A lot of effort has been made to improve the user interface.

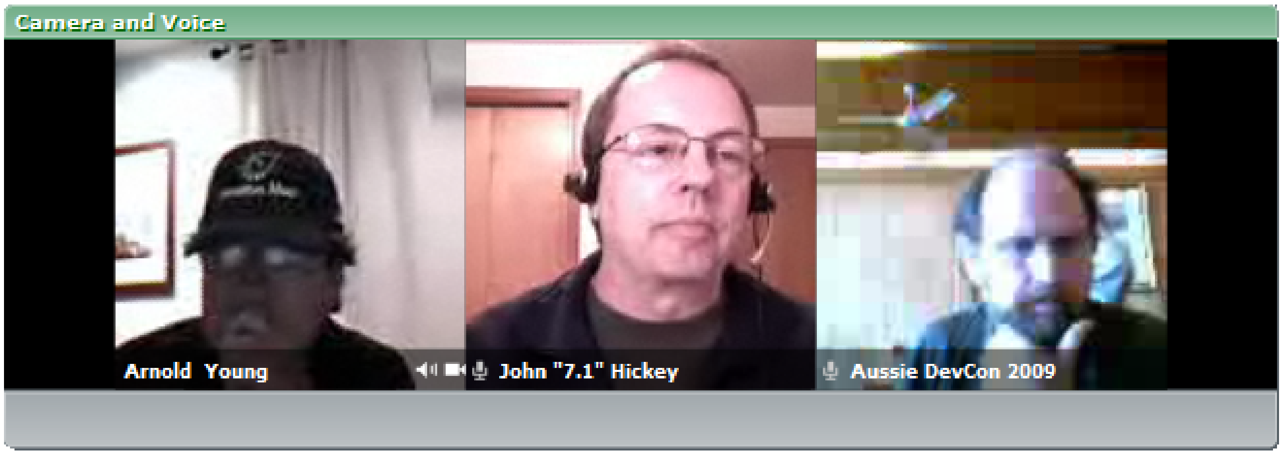


Figure 1. A sinister looking Arnold Young, John Hickey and Geoff Spillane, DevCon organizer

Bob Z has indicated that conference attendees may be able to get the first public release of 7.1.

David Griffith stepped up to the webcam to give his impressions, but unfortunately that bit isn't on the ClarionLive! recording. David had a good impression of the new report writer, which in his view has all the features that you'd [probably] get in List & Label. Other than that he didn't see a lot of new features in 7.1, rather improvements here and there. The source editor (if I understood David correctly) has also been improved.

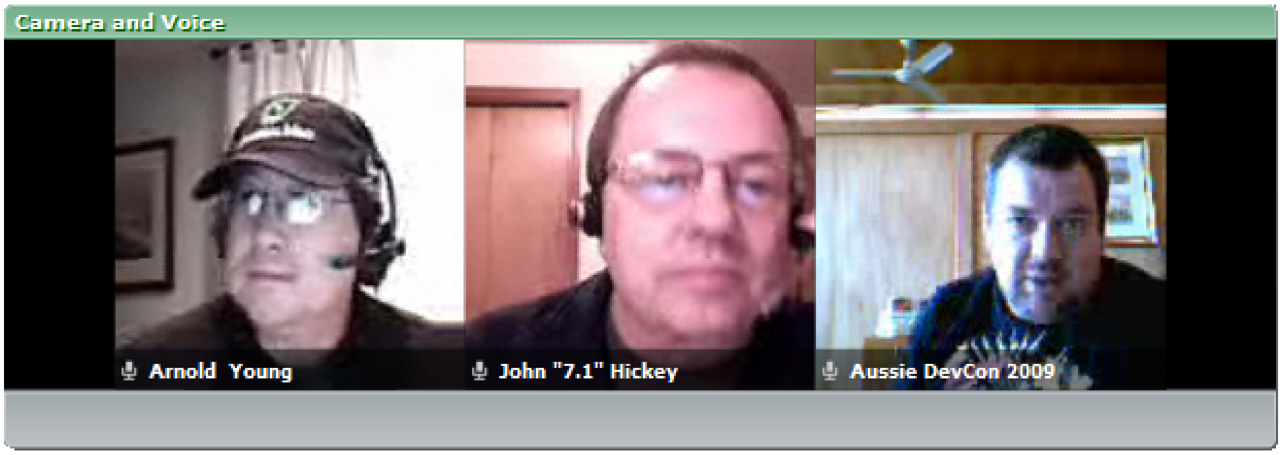


Figure 2. David Griffiths on the right.

Ted Steward pointed out that in reality 7.1 is what 7 should have been. There are improvements to usability, menu options have been moved around, there's a lot more consistency. The window previewer is there.

Figure 3. Bob Foreman and Ted Steward on the right, with Geoff Spillane in the background

The new report writer is based on the DevExpress reporting controls, which SoftVelocity has licensed. Ted has been working with DevExpress reporting for the past year in .NET, and says the DevExpress product far exceeds the capabilities of the old C6 report writer. SQL developers in particular "are going to absolutely love it."

As for the current level of integration of DevExpress into the Clarion toolset, Ted pointed out that it's difficult to know without having any hands-on experience.

Bob Foreman stepped up with a clarification on the report writer licensing. He's "almost 100% sure" the licensing absolutely will be the same for the new report writer, and recommends anyone concerned about this read Bob Z's blog post.

Bob Foreman confirmed the new report writer will be include in/with 7.1. He's already written some scripting for the new report writer. There are events to which you can attach scripting, you can add your own functionality, use sub reports, graph views, pivot grid controls.

Bob does expect that the 7.1 beta will have the report writer. Note the term "beta" - it does seem clear that with the new functionality in 7.1 there will be a beta process rather than a direct release.

Bob also alluded to some "really cool news" which Z will unveil later in the week. Bob wouldn't say any more, but word on the ClarionLive! chat is that Diego has been using the .NET AppGen, and the hope is that Bob Z will give a demo. But that hasn't been confirmed.

ClarionLive! is providing a daily live video feed from the conference, now at 6:30 p.m. Pacific time. You can also view previous reports. If you're doing (or planning) any web development in Clarion you will definitely want to watch Bruce Johnson's NetTalk 5 preview.

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

Reader Comments

Add a comment

Clarion Magazine

RSSBuilder Again

by Steven Parker

Published 2009-10-14

A little while ago, I wrote about my experience creating my first brand new application in Clarion 7 (build 5768). The application read a TPS file and created an XML file. Specifically, the XML file contains the information for an RSS feed.

Before continuing, there are some RSS references, in addition to those listed by Dave Harms in "Creating An XML RSS Web Site Summary With Clarion 6 (Part 2)," that I found very helpful:

- Introduction to RSS: WebReference.Com

and

- Search Engine Watch

I found the later reference especially helpful. If you are thinking about creating RSS files, I commend these sites to your attention.

I had several problems with RSSBuilder.APP. Some were clearly of my own making, some clearly were not and some were not assignable.

These problems were:

1. The app was SDI
2. XMLGenerator's root tag issue
3. Parsing of file data
4. Need for a window preview
5. The ongoing Mask bug

SDI Menu

For the life of me, I cannot remember why I created the app's main menu as SDI. Perhaps I was thinking that SDI would limit me, reign me in, in some helpful way. I would not be able to run a report while I had the source data file open. This, theoretically, would ensure that whatever was *in* the file would be in the report and that a report could not be run while data entry was in process.

I quickly discovered that I didn't like SDI. I didn't like opening a browse and seeing the menu bar suddenly go gray (disabled). I didn't like windows displaying outside the main window. I didn't like that I couldn't open my articles and sites browses simultaneously. In other words, I hadn't really thought the design through.

Because there weren't many menu items, deleting the Menu and creating a Frame was not a lot of work. In this case, the change template type option - something with which I've had less than satisfying results in the past - worked nicely.

From the procedure properties screen, press the ellipsis next to the "Template" prompt (Figure 1).

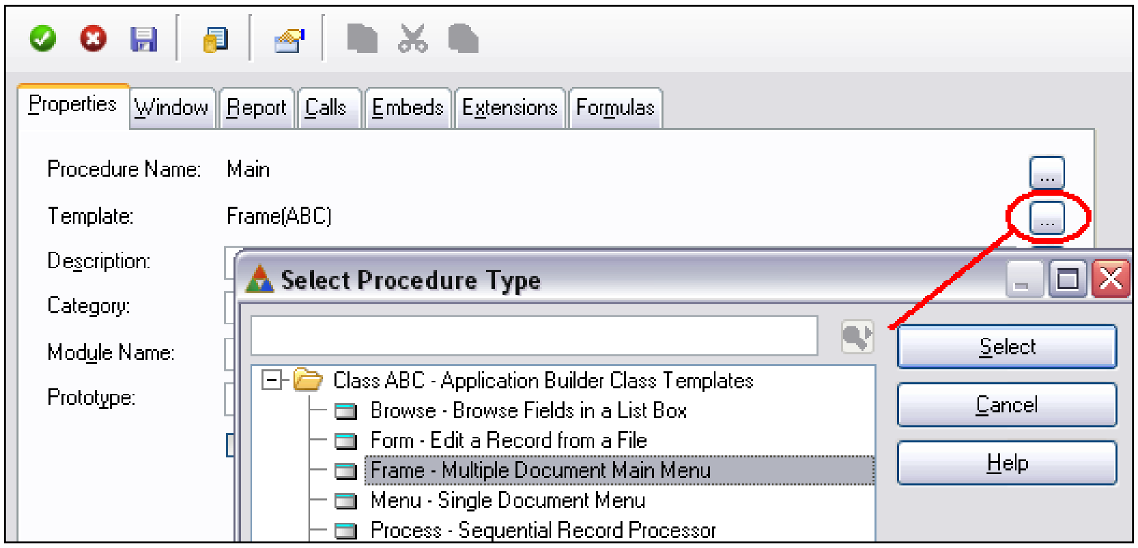
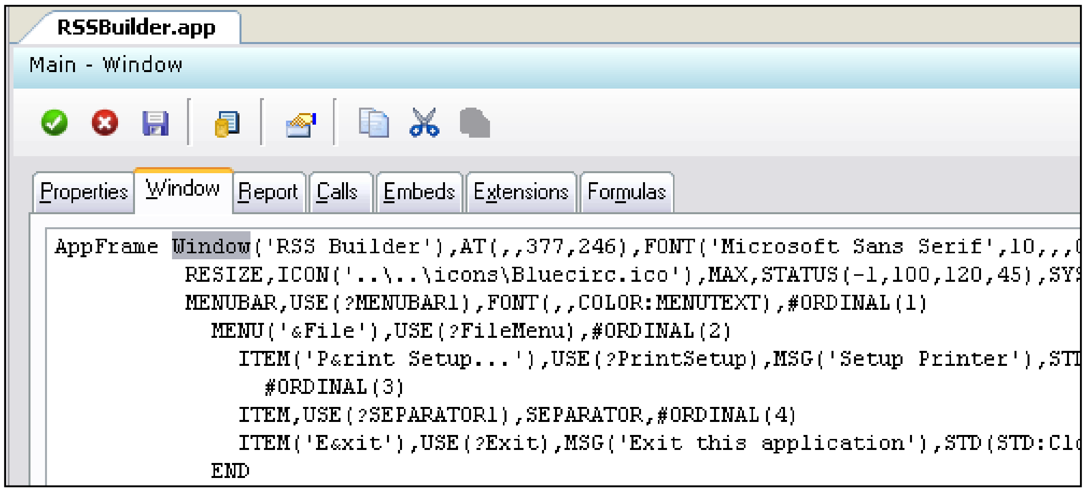

Properties
Window
Report
Calls
Embeds
Extensions
Formulas
Procedure Name: Main
Template: Frame(ABC)
Description:
Category:
Module Name:
Prototype:
Select Procedure Type
Class ABC - Application Builder Class Templates
Browse - Browse Fields in a List Box
Form - Edit a Record from a File
Frame - Multiple Document Main Menu
Menu - Single Document Menu
Process - Sequential Record Processor
Select
Cancel
Help

Figure 1. Change template type

Then select the new template type from the pop-up list.

In this case, one other change is required (Figure 2).



```
AppFrame Window('RSS Builder'),AT(,,377,246),FONT('Microsoft Sans Serif',10,,,
           RESIZE,ICON('..\..\icons\Bluecirc.ico'),MAX,STATUS(-1,100,120,45),SYS
           MENUBAR,USE(?MENUBAR1),FONT(,,COLOR:MENUTEXT),#ORDINAL(1)
             MENU('&File'),USE(?FileMenu),#ORDINAL(2)
               ITEM('P&rint Setup...'),USE(?PrintSetup),MSG('Setup Printer'),STD
                 #ORDINAL(3)
               ITEM,USE(?SEPARATOR1),SEPARATOR,#ORDINAL(4)
               ITEM('E&xit'),USE(?Exit),MSG('Exit this application'),STD(STD:Clo
             END
```

Figure 2. Main Menu window structure

The "Window" attribute has to be changed to "Application." Add the MDI attribute as necessary to the called procedures' WINDOW structures and the "Start" option to procedure calls from the main menu, and I'm done.

Thread Limiting

Changing to an app Frame would allow my articles browse, for example, to be Started multiple times. Therefore, the articles update form could be opened multiple times. *This* is something I don't want.

I've been using a thread limiter template to limit a procedure to a single (non-look up) instance. This template was written by Jim DeFabia in CW2003 and has worked correctly for me in every version of Clarion since. (The template is available in Jim's March '98 Dr. DePhobia column in the Clarion Online archives here at Clarion Magazine.)

Changing the Menu to a Frame provided the opportunity to see if Jim's template still works.

It does.

If you check the code generated by this template, you will note that it is not, strictly, thread safe. A global variable holds a flag as to whether or not a particular procedure is running (there is one such variable added for each procedure to which this code template is added). Strictly, I should wrap the reading and writing of this variable in a Critical Section.

Of course, this would make the template a bit more complex. The Editorial Board of Clarion Magazine discussed this issue several years ago and, while there was general agreement that it is not technically thread-safe, the likelihood of a user being able to double start a single procedure quickly enough to defeat the template was ... minimal. Purism did not win the day (or, at least, was not motivating enough to make me change the template). This is a rarity at the Editorial Board.

Adding Attributes to Root Tags

The problem, here, is that SV's XMLGenerator class does not allow attributes on root tags.

In the XMLGenerator class, the root tag is written as part of the .OpenDocument method. This method, as designed, takes a single parameter, without attributes.

I want to write a line like:

```
<rss version="0.91">
```

but cannot. If I try to pass 'rss version="0.91"' to .OpenDocument, the class writes:

```
<rss>
```

My guess is that the XMLGenerator class is taking a CString parameter. Therefore, it truncates at the first null character. The workaround, created by Dave Harms, is:

```
Myrss.OpenDocument('rssxxxxxxxxxxxxxxx')
```

which writes:

```
<rssxxxxxxxxxxxxxxx>
```

Once all the XML is fully written, following Dave's lead, I loop through the XML file (it is "just" an ASCII file). I look for "<rssxxxxxxxxxxxxxxx>" and replace it with the desired code:

```
If text = '<rssxxxxxxxxxxxxxxx>'
  Myascii.Replace('<rss version="0.91">')
```

This code uses Konrad Byers' AnyASCII class. AnyASCII provides a "replace" method. If I put a generic ASCII file in my dictionary, I could use the ASCII driver and loop through the records. In that case:

```
If text = '<rssxxxxxxxxxxxxxxx>'
  MyASC:DataLine = '<rss version="0.91">'
  Put(MyASCIIfile)
```

should do the job.

The key to both code snippets is that the ASCII driver can safely replace (PUT) only the same number of characters as it has read (see the online docs for the ASCII driver). This is the reason for the initial write of 'rssxxxxxxxxxxxxxxx'. It is exactly the same length as the final output, 'rss version="0.91"'.

Is this a bug or is the XMLGenerator class simply not designed to handle root tag attributes? I suspect the later. But, given that the class does not do what I want, I do have a workaround. Normally, that is enough for me. Indeed, if I had not had other problems with the XML creation, I would have been perfectly content to stop with a functioning workaround.

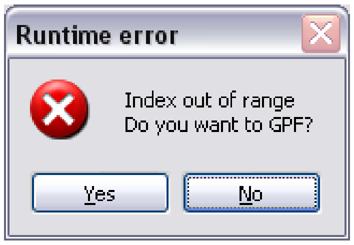
In general, given a choice between waiting for a bug fix and implementing a workaround, I'll go with the workaround. The only potential downside to this, and it is a significant downside, is that when the underlying bug does get fixed, I can end up with very difficult to trace misbehaviors in my apps.

I am specifically thinking of the early days of Clarion for Windows when an ABC open sometimes didn't open a file (yes, defer open *was* off). I often ... supplemented the Open method with a hard open. Later, a bug in the counters was fixed - this was the cause of the original problem - and, voilá, the file could not be properly closed.

The moral of the story? Get your code out the door but note where you've implemented a workaround. Check release notes for new release and match to your list of workarounds.

CData

The other problem that I had was that the XMLGenerator class was throwing errors on invalid index values when writing out the file. There was no helpful information in the message (see Figure 3, below). I finally tracked it down to the use of HTML reserved characters in my article titles and descriptions. It seems that the XMLGenerator class does not escape reserved characters as I would expect it to. (My "oh so subtle" approach was to substitute short dummy strings containing no special HTML characters. When I did this, the XML generated without error. Hence, my *inference* that the XMLGenerator class was not escaping these characters correctly.)


Runtime error

Index out of range
Do you want to GPF?

Figure 3. Error on writing XML

I solved the parsing issue by telling XMLGenerator to treat my data as CData ("Character Data"):

```
Myrss.AddTag('title',Clip(ART:Title),1,'item')
```

CData is not parsed by the XML reader/writer. In other words, CData is left alone, treated as a string literal and simply written into the XML output.

But, with these two issues, I went in search of a less kludge-like solution.

Beyond kludges

The solution to both problems is Robert Paresi's iq-XML (freeware available either at my download center or on Mr. Paresi's Clarion page).

iq-XML is a very full featured XML writer/reader. It is so full featured because Mr. Paresi developed it for his own business needs. That is, iq-XML is a creation of necessity.

In this case, I was only interested in the "write" functions. Robert provides methods to add attributes to a tag at any level:

```
XML:CreateParent('rss')
XML:CreateAttribute('version','0.91')
XML:AddParent
```

This solved my problem with adding attributes to root tags (I had no need to apply the kludge to the closing tag as iq-XML provides (optional) automatic closing of tags - very handy, nothing beats not having to write code).

Note that XMLGenerator makes adding a tag a single method call (see the AddTag call, above). iq-XML makes it a multi-step process. However, not only is this more like the style of code I would write myself in a Process, it allows multiple manipulations (adding multiple attributes) before committing the write. This also makes conditionally adding or setting an attribute easier.

```
x# =  Instring('<13,10>',ART:Description,1,1)
if x# <> 0
  x# = 1
end
XML:AddElement('description',ART:Description,x#)
```

This code writes ART:Description as CData if carriage return/line feed is found in ART:Description (I do want to leave the carriage return/line feeds in the displayed page but normal parsing would remove them). Otherwise, the field is simply parsed. Yes, iq-XML correctly wrote all my file fields, without error (and that is another reason I'm convinced there's an issue in XMLGenerator).

iq-XML has one method I found *very* convenient. XML:AddComment allows me to add comments/notes to the generated XML. Very nice indeed. Also note that iq-XML is not implemented as classes but as straight procedure calls into a DLL his template adds to your app.

I have little to no expertise with XML (and to fully employ a tool as complete and flexible as iq-XML, a knowledge of XML is recommended). So I read the docs, at least the portion on writing XML files. Then I reread the docs. Robert has structured the procedure calls in iq-XML very much like I write standard Process code, so the process of creating the code was very comfortable. The entire exercise took me less than an hour.

The Path Not Taken

There is, of course, an alternative way to create a clean XML file, in a single pass and without kludges, that I haven't mentioned. That alternative is using the ASCII driver to create and write a file myself.

In this case, I would write the opening and closing tags and the data myself.

I have, in fact, done this before. This code:

```
XF:DataLine =    |
  '<USER_ID>' & Clip(PCCUser) & '</USER_ID>'       &|
  '<COMMAND>' & Clip(TRAN:Action) & '</COMMAND>'   &|
  '<TROUTD>' & CLIP(TRAN:TroutD) & '</TROUTD>'
  ! etc.
Add(xFile)
```

is part of an XML file I use to talk to PC Charge. In another part of this same application, I have a custom hand coded parser to read the XML file returned by PC Charge.

So, having done it before, it is my considered opinion that if someone provides a tool, the third party tool is just so much easier, particularly in the case of iq-XML.

Summary

RSSBuilder now functions much more comfortably, much more to my liking. Since I am the customer, I guess that's pretty important.

However, two issues remain from my original list of complaints: window preview and the Mask attribute. Both are supposed to be addressed in the much anticipated 7.1 release (along with the even more anticipated new Report Writer).

As soon as I get my hands on 7.1, I'll check out preview and Mask and get back to you.

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

Reader Comments

Add a comment

Clarion Magazine

Aussie DevCon Day 3 Notes

by Dave Harms

Published 2009-10-14

During the Tuesday night ClarionLive! Aussie DevCon report David Griffiths clarified yesterday's comment on the editor - the feature that has been enhanced is the Expression Editor.

David indicated that .NET feels the same as anywhere else except you can use Clarion code. There's still lots to get used to, but [Clarion#] is relevant at the moment if you're doing a lot of hand coding.

> **Note:** John asked a question about converting applications from Clarion to .NET, and David indicated they had been doing conversions in the class that day. But this is most likely a reference to the DNet template wizard which was released for C6 shortly after Clarion.NET went into preview release. This wizard does let you generate a Clarion# application from a dictionary, but it doesn't convert existing applications. As well, the resulting code was never presented as anything other than a learning exercise, and should not be taken as an indication of the architectural direction of any new Clarion.NET template chain (for that information, stay tuned for Bob Z's keynote presentation, which hopefully will also be available at ClarionLive!

Pierre Tremblay conducted "a very intensive session" on the standard .NET controls. This is Pierre's first time conducting a seminar like this, and despite English not being Pierre's first language his session was enjoyed by the attendees.

John asked if it will be easy to move to .NET. Pierre responded: "If you are curious enough, and you like writing code, .NET will not be something difficult for you."

Of course, the alternative to writing a lot of .NET code is the upcoming .NET AppGen.

Arnold asked: "What's the compelling reason [to move to .NET]?"

Pierre indicated that it may be more appropriate to write new projects in .NET, although Win32 "will never die." And you can mix and match Win32 and .NET code (see Wade Hatler's series in ClarionMag). There are a great many .NET third party products.

Arnold asked if Clarion.NET will replace the web-enabling tools Clarion developers now use.

Pierre responded with some comments on web services (link) and multi-tier applications. Web services can only be part of an application, but can be a useful part of a WinForms application. There are many ways to structure .NET applications, and a .NET application is not necessarily a web application. You could have both a WebForms and a WinForms app using the same web services. If you know how to write a class you know how to write a web service (because the IDE actually writes the code that manages the web service bits).

John asked about the different versions of the .NET framework. Pierre pointed out that later releases of the .NET framework build on top of the previous releases - so 3.0 adds classes to 2.0, 3.5 adds to 3.0 and so forth.

> **Note:** The move from .NET 1.1 to 2.0 *did* break a lot of code, but that's the one exception to the rule. Since that time the .NET 2.0 libraries, which form the core of .NET have remained pretty stable.

What things are really different? There is no ACCEPT loop. Everything is an object, so you will need to learn at least a little OOP if you don't know it already. .NET has the CLR, the common language runtime, and any .NET language must conform to the CLR standards.

Bob Foreman indicated that training classes in .NET will follow after Clarion.NET goes gold. The exact format of that training hasn't yet been decided, but it could be in the form of webinars.

Asked about whether SV was using Clarion.NET to create any real-world applications yet, Bob Foreman indicated that there are Clarion devs who have written some Clarion# applications in hand code. It is the minority though; the Clarion way is the AppGen. As now, there will be base class libraries and template sets, which the Clarion third party community can enhance.

Ted Steward was of the opinion that Gus Creces has some Clarion.NET templates in production.

> **Note:** Ted may have meant libraries rather than templates, as indicated on the Handy Tools site.

Bob F says the dev team is "walking on air" and a lot of progress is being made on the .NET AppGen.

In the latest release of Clarion# there are automatic class properties - no setter/getter needed.

John Dunn asked about whether there are plans for Win32/.NET integration built into the Clarion environment. Pierre didn't know about that, but suggested a PTSS entry. John also asked about support for WPF. Pierre pointed out that the main requirement is to have a WPF designer, and the SharpDevelop IDE (which has been blended with Clarion to create the new Clarion IDE) does not have a WPF designer. But maybe Z will have more to say about that.

> **Note:** At one point SharpDevelop did have a WPF designer, but it was removed from the 3.0 release of the product. The designer is schedule to be reintroduced in SharpDevelop 4.0. SoftVelocity has said their license to the SD code includes future releases.

Ted Steward was pumped up about the morning's session. It wasn't so much any one thing, more the sum of many small things and nuances of the .NET framework. A number of the attendees seemed to get a lot out of the toolbars/menubars discussion. As of the morning there hadn't been any discussion of COM/OCX controls in the Clarion.NET IDE.

On Wednesday the ClarionLive! webinar starts at 6:30 p.m. Pacific time as usual, but come half an hour early for a panel discussion of some of the .NET issues raised in this report.

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

Reader Comments

Add a comment

login | faq

search

CLARION MAGAZINE'S CLARION FAQ

Questions · Tags · Users · Badges · Unanswered · Ask Question

Recent Questions

active · 0featured · hot · week · month

0 votes 0 answers 0 views

Calculate Nmbr of Years (AGE)

years lapsed age

24m ago Rakesh 1

0 votes 2 answers 6 views

What's the URL for the PTSS system for C7?

ptss clarion7

4h ago Mike Hanson 86●6

0 votes 2 answers 13 views

Date fields in SQL

sql datetime date time convert

16h ago Bruce Johnson 11●2

0 votes 1 answer 8 views

How do you install the Topspeed ODBC driver on a workstation

topspeed odbc

2d ago Rick Martin 116●5

0 votes 2 answers 94 views

RTF issues: inserting RTF from one control to another

rtf rtfcontrol text

nov 6 at 21:44 Community♦ 1●1

1 vote 2 answers 38 views

How can a global template share its settings across multiple APPs?

global extension templates

nov 6 at 13:47 Bjarne Havnen 11●1

Welcome to ClarionMag's Clarion FAQ

A place to ask questions about Clarion programming.

Sponsor

This site is sponsored by Clarion Magazine.

Recent Tags

all tags »

Recent Badges

all badges »

Sponsor

This site is sponsored by Clarion Magazine.

0 votes 0 answers 7 views

Using Das tag on condition to print

ssssdssssssssdsedede

nov 6 at 13:16 Peter Bosire 1

0 votes 2 answers 52 views

Can a window be made to recalculate constantly, as the user is editing fields?

calculate event-accepted event-timer focus prop-touched

nov 6 at 10:57 Mike Hanson 86●6

0 votes 2 answers 41 views

control{prop use} = dynamicfile.column ? How?

dynamic driver

nov 6 at 9:14 Bjarne Havnen 11●1

0 votes 3 answers 45 views

Base 64 decoder ?

base64 decoder

nov 6 at 5:52 Bruce Johnson 11●2

1 vote 4 answers 117 views

Base 64 Encoder ?

base64 encoder

nov 6 at 5:41 Bruce Johnson 11●2

0 votes 2 answers 21 views

How do I add customized hotkeys C7/Clarion.NET IDE?

c7 clarion.net ide-customization hotkey

nov 5 at 16:50 MarkG 1

0 votes 1 answer 39 views

Printing on custom

print-custom-paper

nov 5 at 0:48 Joe Snyder 16

0 votes 2 answers 38 views

How To Create New Odbc Connection If Current Connection Is Gone Before Firing Query

odbc disconnect reconnect

nov 3 at 21:00 Rick Martin 116●5

0 votes 1 answer 43 views

How do you create a sub menu in an ABC browse right click popup

abc browse popup submenu

oct 29 at 17:48 Rick Martin 116●5

---

0 votes 2 answers 48 views

How do you apply the same wallpaper to all of your windows?

wallpaper global template

oct 29 at 14:59 Mike Hanson 86●6

---

0 votes 3 answers 108 views

When do you use TakeCompleted

use of take completed

oct 29 at 10:40 Bruce Johnson 11●2

---

0 votes 3 answers 94 views

What kinds of problems do Clarion apps have with multicore CPUs?

multicore cpu

oct 29 at 10:02 Bruce Johnson 11●2

---

0 votes 1 answer 35 views

Is there a list of TPS error codes?

tps topspeed error errorcode

oct 28 at 17:53 David Harms♦♦ 73●5

---

0 votes 1 answer 21 views

Link Problem on app that used to work

link vista

oct 28 at 16:05 Mike Hanson 86●6

---

1 vote 3 answers 108 views

how do I make a variable in a global dll template available to other dlls ?

templates export

oct 25 at 23:42 Rex Kersley 1

---

0 votes 1 answer 22 views

The “No Sheet” setting on a SHEET causes strange tab positioning in C6. Is there a fix?

sheet nosheet tab run-time properties

oct 25 at 12:53 Mike Hanson 86●6

---

2 votes 3 answers 137 views

How Can I Get A Global Template To Add Code To Every Procedure?

template-language

oct 24 at 15:25 Mike Hanson 86●6

---

0 votes 4 answers 265 views

What kinds of issues (if any) do Clarion apps have with Windows 7?

windows-7 compatibility

oct 24 at 9:08 Zeljko Manjkas 1

---

0 votes 3 answers 178 views

Do I need a commercial license to use MySQL with my Clarion application?

mysql license free

oct 23 at 7:42 Mogamat 1

---

0 votes 1 answer 38 views

Can I delete one table from a Clarion Superfile?

topspeed superfile

oct 22 at 16:52 Rick Martin 116●5

---

0 votes 1 answer 89 views

Why Can’t I View My Clarion 7 Encrypted Files In TopScan?

clarion7 topscan encryption

oct 21 at 23:41 Stu Andrews 25●6

---

1 vote 5 answers 178 views

What changes are there to the template language and AppGen in Clarion.NET?

appgen clarion.net template-language

oct 21 at 17:54 Dennis 11●1

---

0 votes 1 answer 85 views

where did you find the Cla$Start details

clastart

oct 20 at 22:24 Community♦ 1●1

---

0 votes 1 answer 121 views

How do I convert a C7 application back to C6?

txd txa c7-to-c6 backport

oct 9 at 15:35 Richard Rose 1

---

1 vote 1 answer 86 views

How do you execute a procedure by address?

address procedure start

oct 7 at 19:57 Rick Martin 116●5

0 votes 3 answers 138 views

How do I START a procedure by its name?

start

oct 7 at 17:21 MarkG 1

0 votes 2 answers 126 views

Where can I find official announcements about Clarion 7?

c7 clarion7 softvelocity

oct 7 at 13:57 Mark Sarson 11●2

Looking for more? Browse the complete list of questions, or popular tags. Help us answer unanswered questions.

recent questions feed

Some example footer content.

Clarion Magazine

Aussie DevCon Day 4 Notes

Published 2009-10-15

During the Wednesday night/Thursday noon (US vs Oz time) webcast Bruce Johnson picked up the laptop and went on walkabout, showing a bit more of the conference facilities as well as the "downtown" street view. Eden's weather has been cloudy with showers, but even through the webcam it looks like a pretty spot.

News-wise it was a slow day, and there's not much to add to what's already been said. Attendees were on the home stretch of the four day training marathon, and while Bob Zaunere had been seen in the area he wasn't available to take questions, nor would it make a lot of sense for him to do so until after his .NET presentation.

The conference gets underway Friday morning local time with a whale-watching tour. Bruce looked slightly less intrepid than usual at the thought of riding the swells for a few hours, boats apparently not being his thing. Once the whales are out of the way the conference gets underway, with Bob Z's keynote on the architecture of the .NET templates, and a possible (not guaranteed, but possible) demonstration of the .NET AppGen.

Tune in to ClarionLive! at 9:30 p.m. PDT to hear Bob Z's presentation live from Eden.

Reader Comments

Add a comment

Clarion Magazine

Returning Multiple Values

by Steven Parker

Published 2009-10-26

Suppose I have an order entry window (a POS cash register will do as well). I enter or scan an item. I look the item up in inventory. If the item is not in inventory, I display an error message. If it is in inventory, I display the item's price (let's ignore alternate pricing schemes like on sale or customer price groups) and description. Standard stuff.

Now I want the user to enter the quantity (actually, I usually get the quantity first; if there is no quantity, I assume a quantity of 1) and I want to compute the extended price. Since there may be multiple places I want to compute extended price, I create a separate procedure to do the computation and return the extended price. I could also create a class but for my current purpose, this offers no elucidation. In fact, what I'm going to discuss works equally well with either. So, for the sake of simplicity, I'll stick with a procedure call for this article.

Something like this call is what I have in mind:

```
    Q:ExtPrice =  GetExtendedPrice(Q:Quantity,Q:ItemPrice)
```

The prototype for GetExtendedPrice would look like this:

```
    GetExtendedPrice(Long,Real),Decimal
```

(Actually, I would not usually take a Real in the second parameter but a Decimal. But Decimals involve limitations — specifically that Decimals must be passed by address — that I don't want to get into just yet.)

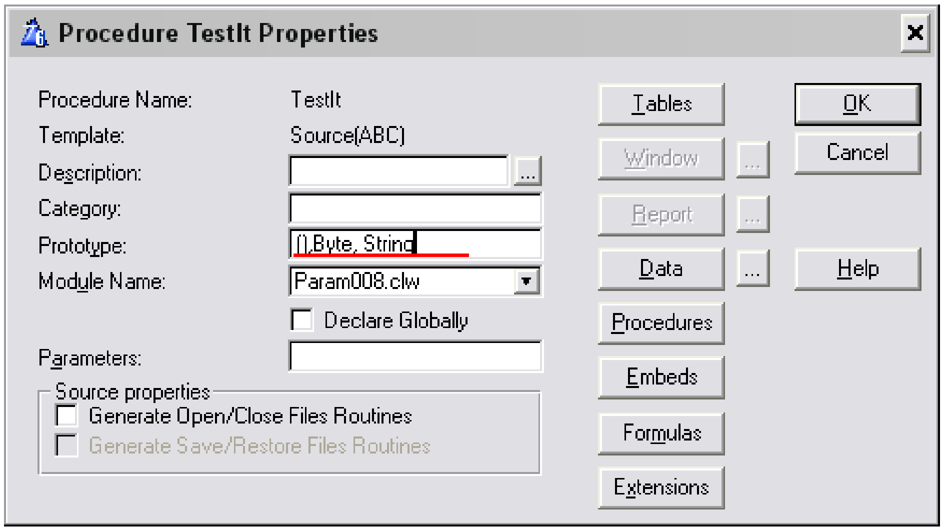
Now suppose I want to get *both* the extended price and the item's tax from GetExtendedPrice.

I need the called procedure to return two values. Luckily, in this case, I do not have to worry about GetExtendedPrice being threaded (you cannot return values, much less multiple values, from a STARTed procedure). There is no reason whatsoever to call GetExtendedPrice on its own thread.

Verboten

The simple answer is that Clarion does not permit returning more than one value from a procedure.

In point of fact, I cannot find such a statement in the documentation for "Procedure" or for "Prototype." But only one value is permitted in the "return value" slot of a procedure prototype. Indeed, one would think that trying to name two data types after the parentheses of a prototype (see Figure 1 below) would almost certain to fail to compile. But it *does* compile.



Procedure TestIt Properties

Procedure Name: TestIt
Template: Source(ABC)
Description:
Category:
Prototype: (),Byte, String
Module Name: Param008.clw
Declare Globally [ ]
Parameters:
Source properties
[ ] Generate Open/Close Files Routines
[ ] Generate Save/Restore Files Routines

Figure 1. Declaring multiple return values in a Source procedure

I cannot for the life of me say how I would get the two values. I presume that the second value, being separated by a comma, is taken as one of the other attributes of a Prototype.

So, it is impossible to return multiple values from a procedure and that is why I am going to show you how to do it.

Basics Again

If you are not entirely comfortable with parameters, please take a moment to read James Cooke's The Nuts And Bolts Of Passing Parameters: Part 1. This articles covers the basics of why to pass parameters, how to prototype procedures, omitted parameters and some issues involved in passing parameters to threaded procedures — a feature available starting in Clarion 6.

I think of parameters as a form of privileged communication. Parameters ensure that the called procedure has precisely the information the called procedure wants and/or needs it to have. Because of the privileged communication, parameters are a better alternative to global data. No threads mashing global variables, no other procedures stepping on the variables, just precision communications. Even when not strictly necessary, knowing that my inventory look up procedure has exactly the right UPC code to look for comforts me. In other words, just as it never hurts to Free a queue, in the immortal words of Ron Eisner, it never hurts to pass parameters.

There are two ways to pass a parameter:

- by value
- by address

Understanding how each of these methods works is important in solving the problem of returning multiple values to a calling procedure.

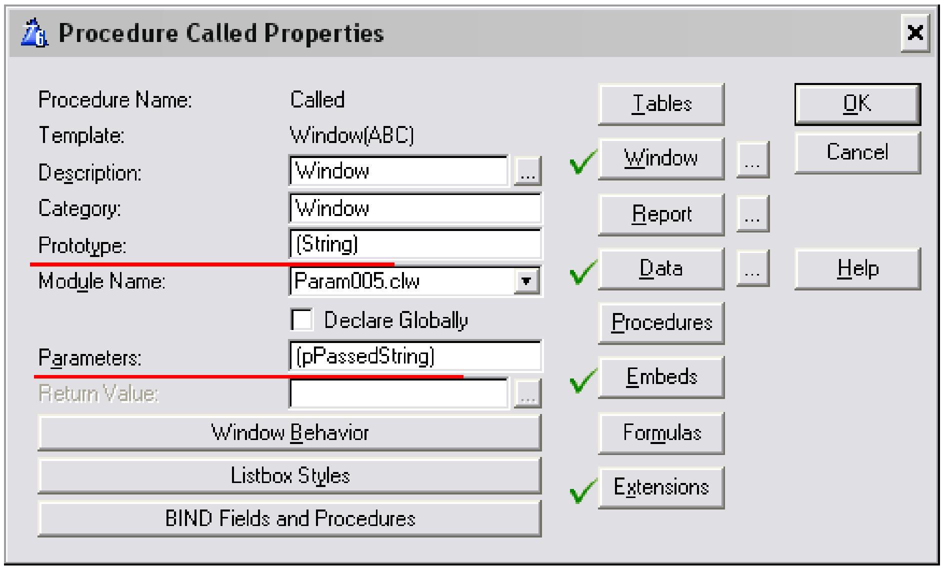
Prototypes

Clarion has long required two pieces of information to implement parameter passing/receiving: a prototype and a parameter list.

The Prototype is a comma delimited list of data types. This list is what the compiler uses to check that calling and called procedures match. (If they don't match, hello "no matching prototype" error.)

The Parameter list is a list of Labels. The Parameter list specifies the variable(s) in the called procedure.

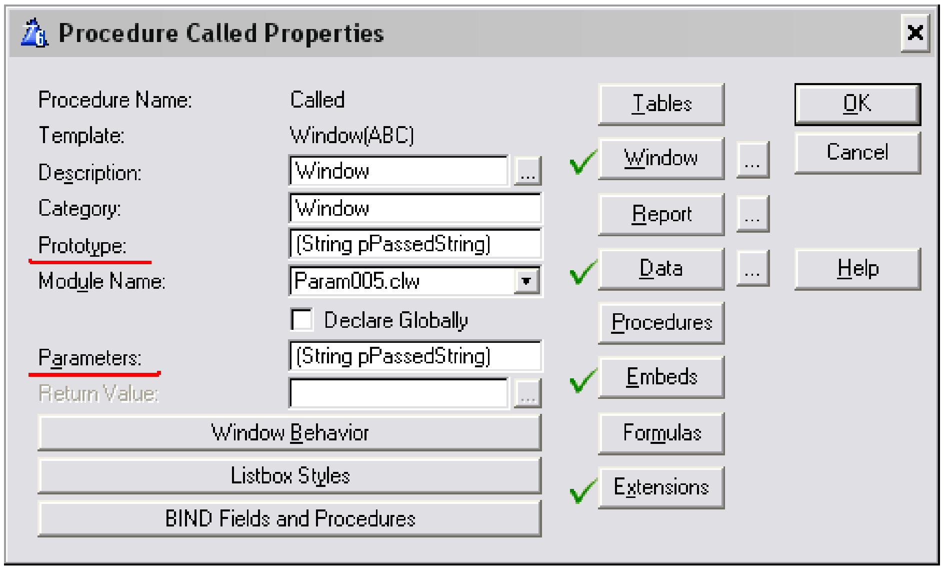
I create the necessary declarations on the Procedure Properties screen:


Procedure Called Properties

Procedure Name:	Called
Template:	Window(ABC)
Description:	Window
Category:	Window
Prototype:	(String)
Module Name:	Param005.clw
Declare Globally	☐
Parameters:	(pPassedString)
Return Value:	

Figure 2. Procedure Properties with Prototype and Parameters, old style

In the 5.x development cycle, a more modern form of specification was introduced:


Procedure Called Properties

Procedure Name:	Called
Template:	Window(ABC)
Description:	Window
Category:	Window
Prototype:	(String pPassedString)
Module Name:	Param005.clw
Declare Globally	☐
Parameters:	(String pPassedString)
Return Value:	

Figure 3. Procedure Properties with Prototype and Parameters, new style

This newer method of specifying the Prototype and Parameter list (the old method continues to work) allows the data type and its Label to be specified at once. However, if I create the Prototype as shown in Figure 3, the Clarion IDE still requires me to complete the Parameter list (instead of just copying it). But, using the new style declaration, I can simply copy and paste the Prototype.

The important part is that these declarations create an implicit data declaration, local to the current procedure:

```
    pPassedString       String
```

Note that the declaration does not specify the string's size. Similarly, numeric parameters will not contain length and precision. In this respect, parameter declarations do differ from standard data declarations.

What this means, however, is that the Parameter Labels can be used in code.

For example, suppose I call GetExtendedPrice as shown above:

```
    GetExtendedPrice(Q:Quantity,Q:ItemPrice)
```

Also suppose the generated code declares GetExtendedPrice as:

```
    GetExtendedPrice(Long pQuantity,Real  pItemPrice)
```

My code could compute the extended price as follows:

```
    Return pQuantity * pItemPrice
```

with no further local data declarations.

The only thing I cannot do using this implicit data declaration is display incoming parameters on a window. This, of course, is because the window designer does not know about my Parameters. To display Parameters, I need to assign them to (local) variables the designer does know about.

By Value

When a parameter is passed by value, the called procedure receives a copy of the original variable. So, for example, if Q:Quantity is 3 and Q:ItemPrice is 4.75 in the calling procedure, the implicit data declarations, logically, look like this:

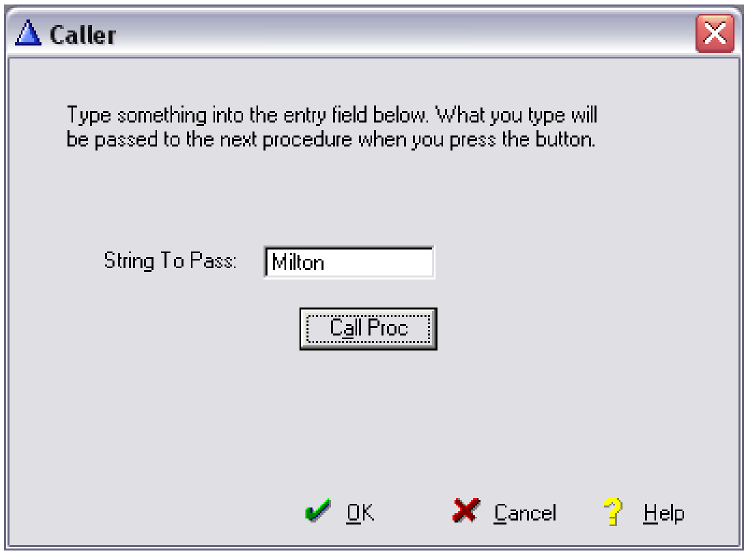
```
    pQuantity      Long(3)
    pItemPrice     Real(4.75)
```

(Well, maybe these implicit declarations aren't so different from normal data declarations after all.)

Those values may be manipulated, changed, in the called procedure but the original variables retain their initial values. Passed "by value," the original variables are never touched. Only the copy can be changed.

In the demonstration app (download-able at the end of this article; this .APP was created in 9056 but a locally compiled EXE is included in case you are still on a pre-6.x version of Clarion), click the "By Value" button. You will get a window in which you can enter a string to pass to the next procedure.

If you enter a value, the "Call Proc" button will be enabled.

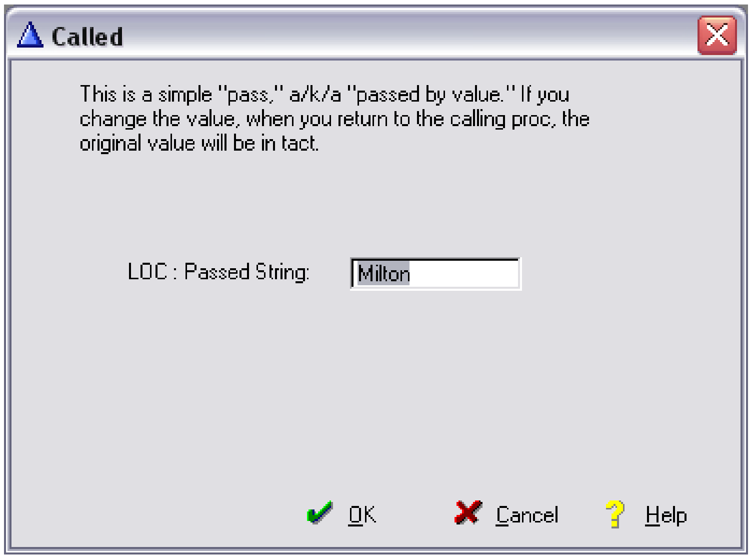


Type something into the entry field below. What you type will be passed to the next procedure when you press the button.

String To Pass: Milton

Figure 4. Set up for passing "Milton" by value

If you then press, the "Call Proc" button, you will see that "Milton" is indeed passed to the called procedure:



This is a simple "pass," a/k/a "passed by value." If you change the value, when you return to the calling proc, the original value will be in tact.

LOC : Passed String: Milton

Figure 5. Parameter assigned to a local variable and displayed

Change the value in the called window. Press "OK." When you return to the calling window, the original value, "Milton," will still be there.

The two procedures for this demonstration are "Caller" and "Called" (Parker's typical inventive procedure naming convention, yes?). They show how to pass by value and demonstrate that passing by value indeed leaves the variable(s) alone.

By Address

Passing variables by address is where life gets interesting. Well, if not "life," then parameter passing....

The online help says:

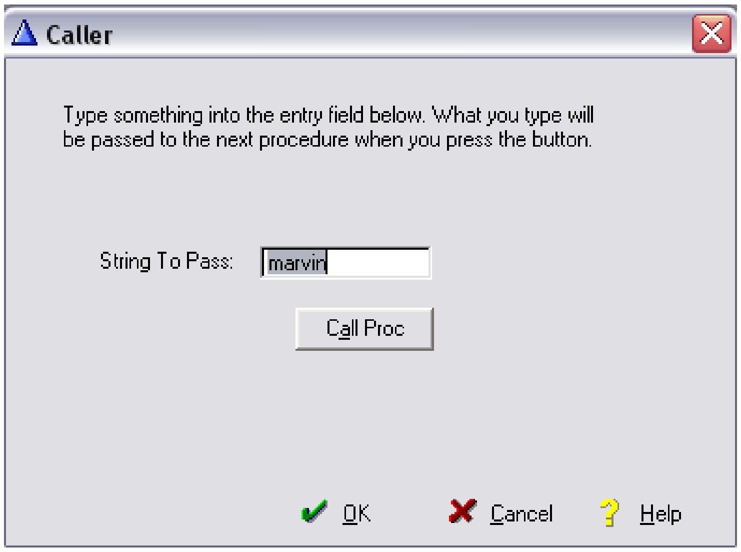
> **Variable-parameters** are "passed by address." A variable passed by address has only one memory address. Changing the value of the variable in the "called" PROCEDURE also changes its value in the "caller." Variable-parameters are listed by data type with a **leading asterisk (*)** in the PROCEDURE prototype in the MAP. [emphasis added]

If you are familiar with the Clarion language, this description should sound familiar. One memory location (i.e., a datum) which is addressable by two names (LABELs). This description sounds very much like OVER:

> The **OVER** attribute allows one memory address to be referenced two different ways

In the demonstration app click the "External" button ("external" is a legacy way of referring to passing parameters by address). You will get a window in which you can enter a string to pass to the next procedure.

If you enter a value, the "Call Proc" button will be enabled.

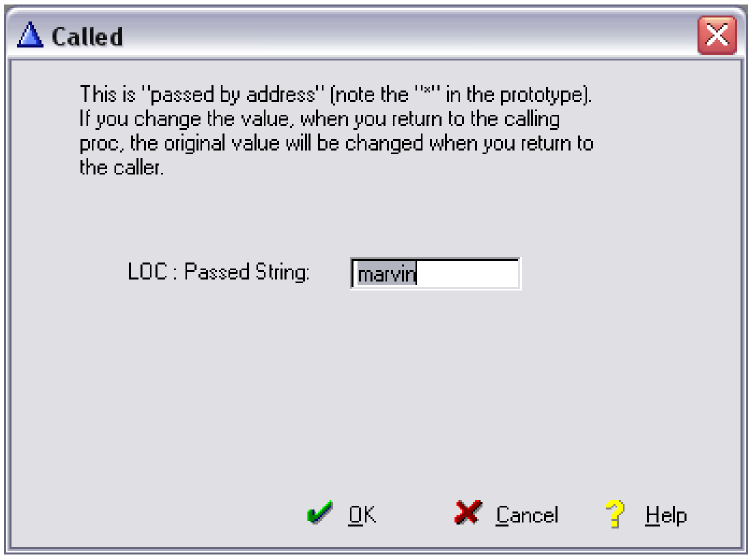


Type something into the entry field below. What you type will be passed to the next procedure when you press the button.

String To Pass: marvin

Figure 6. Set up for passing "marvin" by address

If you then press, the "Call Proc" button, you will see that "marvin" is indeed passed to the called procedure:



This is "passed by address" (note the "*" in the prototype). If you change the value, when you return to the calling proc, the original value will be changed when you return to the caller.

LOC : Passed String: marvin

Figure 7. Parameter assigned to a local variable and displayed

Change the value in the called window. Press "OK." When you return to the calling window, the original value, "marvin," has been changed to whatever you entered in the called procedure.

The two procedures for this demonstration are "Caller2" and "Called2." They show how to pass by address and demonstrate that passing by address indeed allows changing the original variable.

But Wait!

The distinction between value parameters and variable parameters (this appears to be the latest nomenclature for "by value" and "by address") is very important even if you have no interest in returning one value, much less multiple values, from procedures.

In the on line help, look for the topic "Procedure Parameters." It is not immediately apparent but, when you check the types of variables that can be passed in each manner, not all types are listed. For example, Decimals are not in the list of variable types that can be passed by value.

Should you declare a prototype that passes a value parameter and the data type is not appropriate, you will get compiler errors and may find debugging rather... an adventure.

It is for this reason, and so as not to confuse things more than absolutely necessary, that I declared:

```
    GetExtendedPrice(Long,Real),Decimal
```

above. Normally, price would be passed as a Decimal. But

```
    GetExtendedPrice(Long,Decimal),Decimal
```

would be incorrect code. The correct prototype is

```
    GetExtendedPrice(Long,*Decimal),Decimal
```

because Decimals *must* be passed by address.

Returning Multiple Values

In the demonstration app click the "Multiple Values" button. You will get a window in which you can enter three values to pass to the next procedure. Quantity defaults to "1." Price and tax rate are required.

The called procedure is a source procedure and is called when you press the "Compute" button and all three fields have been completed. (I did not take the trouble to enable/disable the button this time. But if you leave one of these fields blank, you will get Clarion standard required field behavior.)

Extended Price, Tax Amount and Total Extended Price are all variables local to the calling procedure.

The code in the called procedure is very simple:

```
    pExtension = pQuantity * pPrice
    pTaxAmount = pExtension * (pTaxRate/100)
    pTotal = pExtension + pTaxAmount
    Return
```

Extended Price is passed quantity times passed price. Tax Amount is Extended Price (computed) times the Tax Rate (passed) divided by 100. Total is the sum of Extended Price and Tax amount (both computed), precisely as one might expect.

But the values, computed in Called3, somehow end up back in Caller3, where they are displayed. Three values computed in one procedure are displayed in another, the caller, and there are no global variables.

In other words, I appear to have returned multiple values from a procedure.

The key, of course, is in the use of variable parameters, parameters passed by address. If you examine Called3, the source procedure that does the computation in the demonstration app, you will find that its Prototype is:

```
    Called3      PROCEDURE(Long pQuantity,*Decimal pPrice,|
                 *Decimal pTaxRate,*Decimal  pExtension, |
                 *Decimal pTaxAmount,*Decimal pTotal)
```

I pass in the quantity, price and tax rate, the first three parameters. Note that the Decimal variable are passed, as required, by address. I also pass in the three "result" fields: LOC:Extension, LOC:Tax and LOC:TotalExension. All of the result fields are passed by address.

The call looks like this:

```
    Called3(LOC:Quantity,LOC:UnitPrice,LOC:TaxRate,|
      LOC:Extension,LOC:Tax,LOC:TotalExtension)
    ThisWindow.Reset(1)      ! ensure display is refreshed
```

Because the result fields are also variable parameters, the computations in the code shown above change the contents of the original variables.

Et violá, I have returned three values!

But Wait!

When I look in the on line help, Prototype Syntax, I see the following:

```
name    PROCEDURE [(parameter list)] [,return type] [,calling convention] ↵
  [,RAW] [,NAME( )] [,TYPE][,DLL( )][,PROC] [,PRIVATE] ↵
  [,VIRTUAL] [,PROTECTED] [,REPLACE] [,DERIVED]
```

which seems to imply that a procedure that returns a value should be prototyped something like:

```
    MyProc(),Long
```

or

```
    MyProc(Long,String),String
```

Prototypes like these create procedures that can be used in expressions. Called3 (prototype and sample call, above) cannot be called in an expression. Called3 does not have a data type after the close parenthesis.

That is certainly true. But that does not mean that Called3 isn't returning values to Caller3. What it does mean is that "return a value" doesn't mean simply what is described in the documentation.

Summary

Well, in fact, a procedure that does specify a return value in the "standard" way can also be called outside an expression. Just add the PROC attribute and both

```
    RetVal = MyProc()
```

and

```
    MyProc
```

are syntactically correct. James Cooke calls these "optional return values" (The Nuts and Bolts of Passing Parameters: Part 2). Didn't realize that a return value, like a parameter, could be optional, did you? So, procedures that do return values, in the standard way, don't necessarily have to be called as if they do....

Returning multiple values in this way does not look like "standard" Clarion. At least, it does not look like standard Clarion when you think about "returning a value." Or does it?

Know the tool. If your syntax isn't illegal... it may just do the job.

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

Reader Comments

Posted on Wednesday, October 28, 2009 by Philip Cumpston

Thank you.
Clear and helpful as usual. I resorted to putting info directly into a record and updateding the record, then having to refresh the queue. This is simpler. Do ypou actually have to pass the address in the function, if you know the variable name and it is active in the calling procedure?
Phil

....

Posted on Wednesday, October 28, 2009 by Steven Parker

Thanks for the kind words.

Yes, you have to pass by address. If you don't, you will work with a copy of the variable (pass by value) and the original (real) variable will not be changed -- unless you use globals (but, of course, the whole point of parameters and return values is to avoid globals and the problems they introduce).

....

Posted on Thursday, October 29, 2009 by Philip Cumpston

Sorry, didn't make myself clear. If you know the name of the variable, can you put a new value into the address of the variable without passing it to the function/ procedure.

i.e., something along these lines.....

e.g. getSomething(var1)

and in the getSomething procedure....

*newInfo = var1 * 4000 + *anothervariableLocatedInTheCallingRoutine

...and have newInfo changed without returning anything.

Of course, this kind of coding is not self-documenting and could lead to a very messy problem for debugging down the track, so I would think it adviseable to make sure everything is passed in the way you describe, just to make sure one knows what one did when looking at it after a period of time has elapsed.

I have enjoyed your contributions to the Clarion Mag. Please keep them coming.

Phil

....

Posted on Thursday, October 29, 2009 by Steven Parker

Got you.

Yes, can be done. If those other variables, not passed in, are in scope. I.e., are declared global or module or are declared STATIC and are available to the called procedure.

Yes, you're right, not, in my opinion, a good practice.

....

Posted on Thursday, November 05, 2009 by Johan Rademan

Great article, great detail.

I find passing a group work really well too, many variables can be passed and the code looks clean.

Add a comment

Clarion Magazine

The Problem With Embeds, Part 2: Extracting And Testing Code

by Dave Harms

Published 2009-10-31

In the introductory article to this series I said that most Clarion developers use embed points the wrong way, and by doing so they make their applications more difficult to maintain, test, debug and document. Almost every Clarion developer has done that; I've done it too. In these articles I intend to show how you can improve your code base by taking the majority of that code *out of the embed points*.

I'll be working through a number of examples, including the Invoice app which ships with Clarion. But in this article I'll focus on the TXA embed parser example I introduced in Part 1.

In any analysis of embed code, and how it might be better deployed, the first thing you need is a convenient way to look at just your embeds. That's not so easy, because, embeds being embeds, they're sprinkled throughout your application.

Really you have two options: You can browse it in the embed list or in the embeditor, or you can use a tool to extract just the embeds so you can look at them without the distraction of the generated code.

About TXAs

As far as I know, there's only one way to programmatically extract the embed code from an APP. First, you have to export a TXA, which is an import/export text version of the information contained in an APP file. And second, you to parse the TXA to get the embeds, which can be a bit of a pain as the TXA format isn't documented.

A couple of years I wrote a TXA parser to do just that task, in support of an article series on the most popular embed points. So it seemed natural, when I needed a way to extract embed points for this series, to revisit that code.

Unfortunately, that code isn't very pretty. Most of it is contained in just a couple of embed points. In the TakeAccepted method's data section there are some declarations:

```
x              long
vars            group,pre()
procname         string(200)
procFromABC      string(60)
procCategory     string(60)
embedname        string(60)
embedPriority    long
embedParam1      string(200)
embedParam2      string(200)
embedParam3      string(200)
whenLevel        byte
                end
dumptrace      byte(0)
LastProcName  like(procname)
lastEmbedName string(500)
currembedname string(500)
```

And then a little later on in TakeAccepted, at an embed that's called when the user presses the Import button, the TXA gets parsed. That code loops through the records in a previously created queue of TXA files (txaq):

```
?progressvar{prop:rangehigh} = records(txaq)
setcursor(cursor:wait)
loop x = 1 to records(txaq)
  get(txaq,x)
  ?progressVar{prop:progress} = x-1
```

```
clear(ema:record)
EMA:TXA = txaq.name
Access:EmbedApp.Insert()
EmbedApp{prop:sql} = 'select last_insert_id()'
next(EmbedApp)
! Add the queue header record
access:TextFile.Close()
 GLO:TextFileName = txaq.name
access:TextFile.Open()
Access:TextFile.UseFile()
set(TextFile)
ProcName = ''
state = 0
lineNo = 0
clear(procname)
clear(lastprocname)
clear(lastembedname)
clear(currembedname)
LOOP
   next(TextFile)
   if errorcode() then break.
   dumptrace = false
   lineNo += 1
   CASE state
   OF 0 ! search for the start of a module or procedure, or an embed
      if sub(txt:rec,1,11) = '[PROCEDURE]'
         clear(vars)
      state = 10
    elsif sub(txt:rec,1,8) = '[MODULE]'
      clear(vars)
      procName = '[MODULE]'
    elsif sub(txt:rec,1,7) = 'EMBED %'
      embedName = sub(txt:rec,7,len(txt:rec))
      state = 30
    elsif sub(txt:rec,1,8) = '[SOURCE]'
      state = 50
    end
  OF 10 ! get procedure name details
    if sub(txt:rec,1,4) = 'NAME'
      procName = sub(txt:rec,6,len(txt:rec))
      state = 11
    end
    do CheckForMissedEmbed
  OF 11
    if sub(txt:rec,1,8) = 'FROM ABC'
      procFromABC = sub(txt:rec,10,len(txt:rec))
      state = 12
    end
    do CheckForMissedEmbed
  OF 12
    if sub(txt:rec,1,8) = 'CATEGORY'
      procCategory = sub(txt:rec,11,len(clip(txt:rec))-11)
    end
    state = 0
    do CheckForMissedEmbed
  of 30 ! Look for a first embed parameter
    if sub(txt:rec,1,11) = '[INSTANCES]'
      state = 41
    elsif sub(txt:rec,1,8) = '[SOURCE]'
      state = 50
    end
  of 41 ! Get first parameter
    if sub(txt:rec,1,6) = 'WHEN '''
      embedParam1 = sub(txt:rec,7,len(clip(txt:rec))-7)
      WhenLevel = 1
      !db.out('whenlevel=' & whenlevel)
    end
    state = 42
    do CheckForMissedEmbed
  of 42 ! Look for a second embed parameter
    if sub(txt:rec,1,11) = '[INSTANCES]'
      state = 43
    elsif sub(txt:rec,1,8) = '[SOURCE]'
      state = 50
    end
  of 43 ! Get second parameter
    if sub(txt:rec,1,6) = 'WHEN '''
      embedParam2 = sub(txt:rec,7,len(clip(txt:rec))-7)
      WhenLevel = 2
    end
    state = 44
    do CheckForMissedEmbed
  of 44 ! Look for a third embed parameter
    if sub(txt:rec,1,11) = '[INSTANCES]'
      state = 45
    elsif sub(txt:rec,1,8) = '[SOURCE]'
      state = 50
      !db.out('found PRIORITY')
    end
  of 45 ! Get third parameter
    if sub(txt:rec,1,6) = 'WHEN '''
      embedParam3 = sub(txt:rec,7,len(clip(txt:rec))-7)
      WhenLevel = 3
      !db.out('whenlevel=' & whenlevel)
    end
    state = 50
    do CheckForMissedEmbed
  of 50  ! look for the priority
    if sub(txt:rec,1,8) = 'PRIORITY'
      embedPriority = sub(txt:rec,10,len(txt:rec))
      if lastprocname <> procname
        ! insert new EmbedProc record
        clear(EMP:record)
        EMP:Proc = procname
        EMP:ProcFromABC = ProcFromABC
        EMP:ProcCategory = ProcCategory
        EMP:EmbedAppID = EMA:EmbedAppID
        Access:EmbedProc.Insert()
```

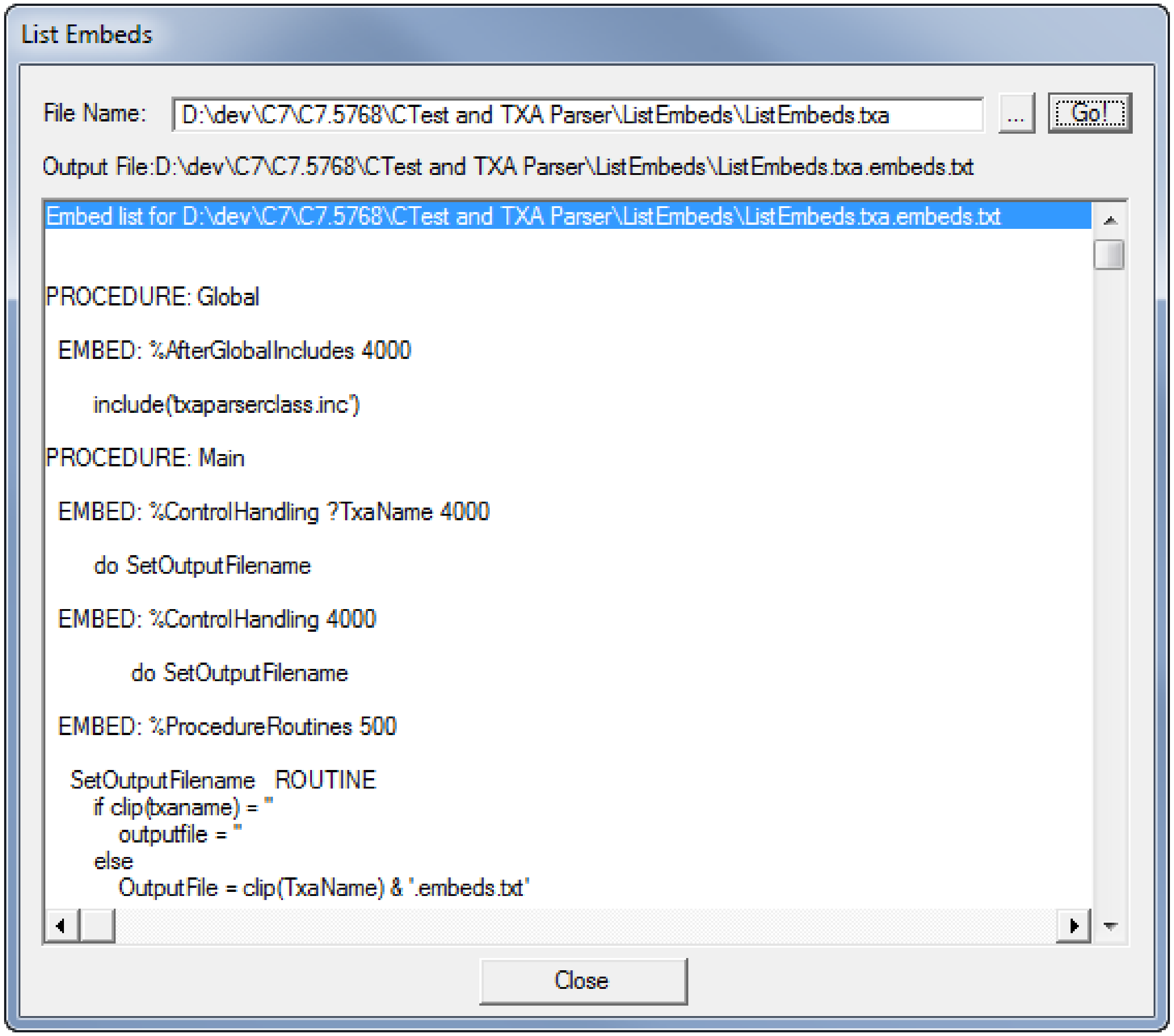
```
            EmbedProc{prop:sql} = 'select last_insert_id()'
            next(EmbedProc)
            lastprocname = procname
          end
          ! Add the embed record
          currEmbedName = clip(embedName) & clip(embedparam1) |
            & clip(embedparam2) & clip(embedparam3) & embedpriority
          if currEmbedName <> lastEmbedName
            lastEmbedName = currEmbedName
            EMB:EmbedProcID = EMP:EmbedProcID
            EMB:Embed = EmbedName
            EMB:Param1 = EmbedParam1
            EMB:Param2 = EmbedParam2
            EMB:Param3 = EmbedParam3
            EMB:Priority = embedpriority
            access:Embed.Insert()
          end
          state = 51
         end
      of 51
        state = 60
        do CheckForMissedEmbed
      OF 60  ! capturing embed
        ! Quit when [END] encountered
        if sub(txt:rec,1,1) = '['
          if sub(txt:rec,1,5) = '[END]'
            case WhenLevel
            of 3
              WhenLevel = 2
              embedParam3 = ''
            of 2
              WhenLevel = 1
              embedParam2 = ''
            of 1
              WhenLevel = 0
              embedParam1 = ''
            end
            state = 0

          elsif sub(txt:rec,1,8) = '[SOURCE]'
            ! look for another embed under this [EMBED] point
           state = 50
          else
            ! could be we're done
            state = 0
          end
        elsif sub(txt:rec,1,6) = 'WHEN '''
          case WhenLevel
          of 0
            ! get the first param
            embedParam1 = sub(txt:rec,7,len(clip(txt:rec))-7)
            WhenLevel = 1
            state = 42
          of 1
```

```
          ! get the second param
          embedParam2 = sub(txt:rec,7,len(clip(txt:rec))-7)
          WhenLevel = 2
          state = 50
        of 2
          state = 50
        end
      else
        ! write embed buffer
      end
    else
      do CheckForMissedEmbed
    END
  END
  Access:TextFile.Close()
end
?progressVar{prop:progress} = records(txaq)
setcursor()
```

In the downloadable C7 zip, have a look at the ImportTXAs procedure in Embeds.app for the complete source (C6 version available on request).

What I had in mind for my new utility was something more along the lines of Figure 1.



D:\dev\C7\C7.5768\CTest and TXA Parser\ListEmbeds\ListEmbeds.txa

Output File:D:\dev\C7\C7.5768\CTest and TXA Parser\ListEmbeds\ListEmbeds.txa.embeds.txt

```
Embed list for D:\dev\C7\C7.5768\CTest and TXA Parser\ListEmbeds\ListEmbeds.txa.embeds.txt

PROCEDURE: Global
 EMBED: %AfterGlobalIncludes 4000
     include('txaparserclass.inc')
PROCEDURE: Main
 EMBED: %ControlHandling ?TxaName 4000
     do SetOutputFilename
 EMBED: %ControlHandling 4000
         do SetOutputFilename
 EMBED: %ProcedureRoutines 500
  SetOutputFilename   ROUTINE
     if clip(txaname) = ''
        outputfile = ''
     else
        OutputFile = clip(TxaName) & '.embeds.txt'
```

Figure 1. A utility to display and write the embed list

My original parser's functionality was overkill for this new app; I really didn't need to build up an elaborate database of applications, procedures, and embed points. I just needed a list of embed points. But obviously I needed all of the parsing capabilities, gnarly though the code might be.

Unfortunately, there wasn't any way to use my original code unchanged, primarily because that code didn't actually extract the *embedded source* from the TXA. There was no need to capture the actual embed code since I was just logging embed usage. And I was motivated *not* to store the embed code: I had asked for TXA submissions which could contain sensitive information, and I didn't want to accidentally expose anyone's embed code to public view.

So what are the options? A few come to mind:

- Just cut and paste the source. This is what a lot of us do, and it has the obvious drawbacks of creating multiple versions of the code to maintain. If I find a bug in my parsing code, I'll need to hunt down every place I've pasted a copy and make the change there.
- Put my original procedure in a DLL and call it as needed. In this case my DLL also presents a user interface, so that would result in some UI clunkiness in the app I envisioned in Figure 1, which doesn't need to call yet another window just to do the parsing.
- Put the common source in source files and INCLUDE them. I could even include just portions using labels. The drawback here is that there's no way to know how the various sections of source code might be used, or how any bug fixes to that source might cause unexpected bugs Using INCLUDE statements this way results in an almost complete loss of control over the source code.
- Create a template containing the source code. This certainly helps keep the code in one place, but it has a lot of negatives when it comes to maintaining and testing that code since you have to put the template in an app and generate the code, and then you have to port any changes back to the template.

And there are other problems. Because my original app was tied to a particular data store (a PostgreSQL database), any re-use of that code would have to know the table definitions. Since Clarion only supports one dictionary per app, any apps that used this procedure would either need to use the dictionary or import the tables from that dictionary.

Class or procedure?

So what's the answer? If it's not a template, and not a multi-purpose generated procedure and not INCLUDEd source, what's left? Procedures and classes, that's what. But not just any procedures or classes. You want to write code that has as few dependencies as possible.

Some of the dependencies you want to avoid:

- Files/tables - don't tie your code to a specific database
- Windows/controls - don't tie your code to a specific user interface element
- Other code - keep calls to other procedures/classes to a minimum

The code you *do* write in embed points can wire up the dependencies; keep your procedures and classes as clean as possible.

But that raises another question: when should you use a class, and when a procedure?

In almost all cases a class is preferable to a procedure, in the same way that a procedure is almost always preferable to spaghetti code. A procedure presents a single point of entry and a single result. That's not to say you can't return multiple values from a procedure - you clearly can, as Steve Parker recently showed. But procedures don't have the flexibility of classes.

In fact, a class method is really just a procedure, so using a class already gets you everything a procedure can do. But it also gets you more, because in a class multiple methods can operate on shared data.

I'm not going to go into all the details of how to create a class; I'll cover that in following articles. For now, just keep your eye on the transformation of the embedded code into a class, and don't worry excessively about exactly how it was done.

The class

The first decision I have to make is how to store the embed data. Originally I used a SQL database, but this now appears to be a liability. My parser should use something more transient, which can be converted to a permanent store or just discarded after use. An in-memory data store, in the form of nested queues, fits the bill:

```
TxaEmbedLineQueue   queue,TYPE
line                cstring(1000)
                END
TxaEmbedQueue       queue,type
embedname           string(100)
EmbedLineQ          &TxaEmbedLineQueue
                END
TxaProcedureQueue   queue,TYPE
```

```
ProcName            string(40)
EmbedQ              &TxaEmbedQueue
                END
```

The parser's job will be to populate these queues, so that I end up with a TxaProcedureQueue containing one or more records. Each of those procedure records has one or more TxaEmbedQueue records, each of which has one or more TxaEmbedLineQueue records for each line in a given embed point. With that queue structure in hand I can easily update a database or create a text file, as I see fit.

At first blush this looks like an ideal task for a procedure. Pass in the name of the TXA and an empty queue and get back a filled queue: presto! But here's why I think the procedural solution is almost never a good solution: there's almost always some new functionality you can add which doesn't fit into the existing procedure.

Think about error handling. You can have the parsing procedure return an errorcode if the parse fails for any reason, but what if you want to get a more detailed error response? What if you want to enable tracing or logging? How would you do that in a single procedure call, without burdening the procedure with some obscenely large number of parameters?

In fact, once I began rewriting my parsing code as a class I ended up with rather a lot of methods and a few properties as well:

```
TxaParserClass      CLASS,type,module('txaparserclass.clw'),link('txaparserclass.clw',|
                  _ABCLinkMode_),DLL(_ABCDllMode_)
ErrorMessage          string(500),PRIVATE
ExternalQueue         byte(false)
dbg                 &Debuger,PRIVATE
ProcQ                 &pTxaProcedureQueue
AddNewEmbed             procedure(string embed,string embedparam1,|
                      string embedparam2,string embedparam3,|
                      string embedpriority),private
AddNewProcedure         procedure(string pname),private
CheckForMissedEmbed     procedure(string s,long lineno,long state),private
Construct             PROCEDURE
```

```
RemoveCurrentProcedureFromQueue procedure,private
GetLastError         procedure,string
Parse                PROCEDURE(string s),BYTE
Reset                PROCEDURE
SetDebuger           procedure(Debuger dbg)
SetQueue             procedure(TxaProcedureQueue procq)
Trace                PROCEDURE(string s),private
Write                procedure(string filename),byte,proc
                 end
```

I won't go into all of these methods in detail - you can have a look at the downloadable source if you're interested. Briefly, there are a few private methods to break up the internal code into reusable blocks (you'd use routines in a procedural implementation), and there are some public methods to get the last error, parse the specified TXA, reset the parser, assign a debugging object, specify the particular queue to use, and write the contents of the queue out to a text file.

Now, show me how you'd implement all *that* in a procedure!

I now have a Write method that dumps my embeds to a text file, but I could also create another Write method that would dump the embeds out to a database. I'd have to do this by passing in file and field references, and perhaps FileManager references, but it could be done, and it would l ensure the code remained portable between not just apps but also dictionaries.

Automated testing!

Perhaps more interesting to me than simple reuse is that by moving my code into a class I've made it *testable*. But what does testing really mean, in the Clarion world? Usually it means writing some code in an embed point, running the app, clicking some clicks and watching what happens. That's useful, but it doesn't necessarily tell you what you need to know about the reliability of your core code. And it's tedious.

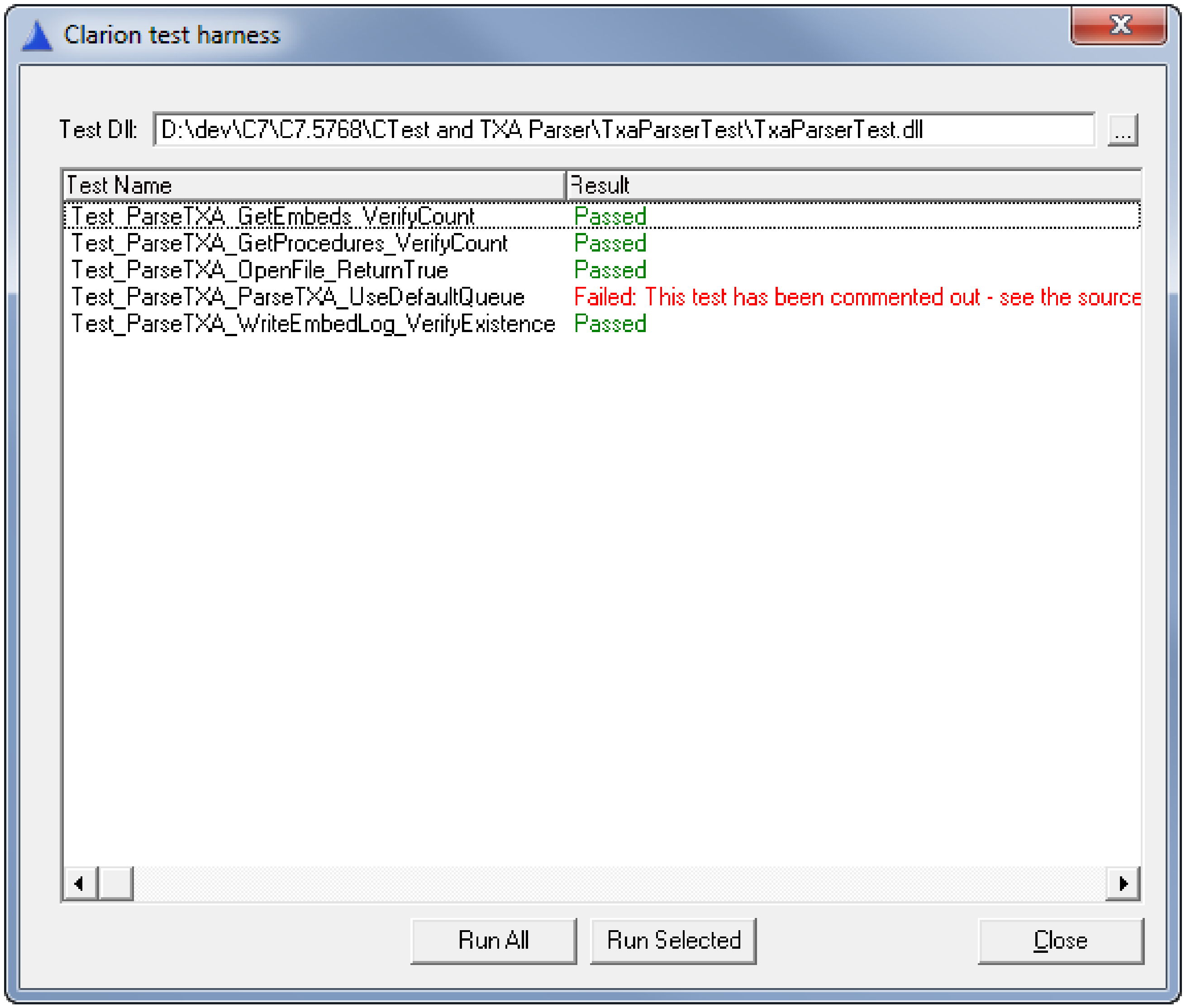
You can take away some of the tedium with automated testing tools, except that historically Clarion hasn't played well with those tools because Clarion apps use custom Windows controls.

But you still won't be testing your app's core functionality in a repeatable way.

Testing is a complex subject, and there are lots of different ways to test application code, but certainly one of the most useful is *unit testing*. The core idea is that you reduce your code to the smallest testable units, and then you run tests on a regular basis. Those tests must be automated; they have to run without user intervention (other than to kick off a series of tests, although you might want a test suite to run automatically on a regular basis, or as part of a build).

Unit testing is commonplace in the .NET world, in part because certain features of the CLR (in particular, reflection) make it fairly simple for testing tools to examine assemblies (DLLs), locate classes and methods marked as test code, run those tests and report on the results.

Automated testing is a bit more awkward in a language like Clarion, but still possible, as Figure 2 shows.


Clarion test harness

Test Dll: D:\dev\C7\C7.5768\CTest and TXA Parser\TxaParserTest\TxaParserTest.dll

Test Name	Result
Test_ParseTXA_GetEmbeds_VerifyCount	Passed
Test_ParseTXA_GetProcedures_VerifyCount	Passed
Test_ParseTXA_OpenFile_ReturnTrue	Passed
Test_ParseTXA_ParseTXA_UseDefaultQueue	Failed: This test has been commented out - see the source
Test_ParseTXA_WriteEmbedLog_VerifyExistence	Passed

Figure 2. Testing the parser class

Testing the parser

Figure 2 is a demostration of two applications, neither of which I've yet discussed. The application that you see running is called (tentatively) CTest, and is a Clarion *test runner*. CTest's job is to load up a specified DLL, search it for test procedures, run those test procedures and report on the results. It's very loosely patterned on the kinds of unit testing applications readily available to .NET developers. I'll have an article describing the inner workings of CTest next month.

The second app involved in Figure 2 is a DLL APP called TxaParserTest. This APP contains some test procedures created using a special version of the Source procedure template.

Here's the code for a procedure called Test_ParseTXA_OpenFile_ReturnTrue. First there are some data declarations:

```
parser                TxaParserClass
q                     TxaProcedureQueue

txa                   string(500)
```

And then there's some code:

```
parser.SetQueue(q)

txa = '..\Invoice\invoice.txa'
if parser.parse(txa)
   tr.Passed = true
ELSE
   tr.message = parser.GetLastError()
END
```

There's some additional code which is automatically generated by the procedure template; I've only shown the embed code I added to create the test. For instance, the tr object is an instance of TestResultT, which is a utility class used to report results back to the test runner (CTest), and tr.Passed defaults to false. Again, I'll get into the details in a future article.

The entire purpose of this test is simply to verify that the parser's Parse method execute successfully, indicating the specified TXA was found and something was done with it.

The Test_ParseTXA_GetProcedures_VerifyCount test verifies that the parser found the correct number of procedures containing embeds. The data is similar to the previous test, so here's just the code::

```
parser.SetQueue(q)
txa = '..\Invoice\invoice.txa'
if parser.parse(txa)
   if records(q) = 13
      tr.Passed = true
   ELSE
      tr.message = 'Expected 13 procedures but found ' |
        & records(q)
   END
ELSE
   tr.message = parser.GetLastError()
END
```

Test_ParseTXA_GetEmbeds_VerifyCount is similar but requires some additional data:

```
expectedEmbedCount          long,dim(20)
x                           long
```

And the code:

```
parser.SetQueue(q)
txa = '..\Invoice\invoice.txa'
if parser.parse(txa)
   expectedEmbedCount[1] = 1
   expectedEmbedCount[2] = 1
   expectedEmbedCount[3] = 2
   expectedEmbedCount[4] = 4
```

```
      expectedEmbedCount[5] = 1
      expectedEmbedCount[6] = 3
      expectedEmbedCount[7] = 4
      expectedEmbedCount[8] = 3
      expectedEmbedCount[9] = 2
      expectedEmbedCount[10] = 1
      expectedEmbedCount[11] = 8
      expectedEmbedCount[12] = 4
      expectedEmbedCount[13] = 2
      tr.passed = true
      loop x = 1 to records(q)
        get(q,x)
        if expectedEmbedCount[x] <> records(q.EmbedQ)
          tr.Message = 'Test failed on index ' & x |
            & ', procname' & q.ProcName & ', count ' |
            & records(q.EmbedQ)
          tr.passed = false
          break
        END
      END
    ELSE
      tr.message = parser.GetLastError()
    END
```

Here's another test, this time called Test_ParseTXA_WriteEmbedLog_VerifyExistence. Here's the code:

```
    txa = '..\Invoice\invoice.txa'
    parser.SetQueue(q)
```

```
if parser.parse(txa)
   if parser.Write('ThisIsATestFileAndCanBeDeleted.txt')
      if ~exists('ThisIsATestFileAndCanBeDeleted.txt')
         tr.message = 'Output file was not created'
      ELSE
         tr.passed = true
      END
   else
      tr.message = parser.GetLastError()
   END

ELSE
   tr.message = parser.GetLastError()
END
```

This test verifies that the parser was able to write a specific test file.

These tests are by no means exhaustive, but they do illustrate the usefulness of extracting code from embed points. Not only can I reuse that code in other applications, I can test the code and gain confidence that it's doing exactly what it should be doing. This is especially important when it comes to modify the code, either because a bug was found (in which case there should be a new test that confirms the bug fix) or because some new feature is needed.

Any time you change code you run the risk of introducing new bugs; having a comprehensive test suite greatly reduces the likelihood of those new bugs going unnoticed. It's also a great way to verify that software upgrades (whether Clarion, or third party products, or even operating systems) haven't broken your code.

Obviously, this kind of code extraction works best with code that isn't tied to the UI and isn't dependent on a particular database, although in most cases, and with TPS files in particular, you can still run automated tests more easily against a database than you can against a user interface. But that's also my point: the code that really gives you application value is *almost always* code that doesn't depend on the UI or on a particular physical database. It may depend on a certain data *structure*, but that structure seldom has to be just a TPS file or only a SQL table or whatever specific implementation you currently use.

Embed code reduction in the embed utility

So what's the end result of refactoring my parser into a class? After exporting the TXA from my ListEmbeds.app utility, I ran that utility against the TXA and got this output:

```
PROCEDURE: Main

  EMBED: %ControlHandling ?TxaName 4000

        do SetOutputFilename

  EMBED: %ControlHandling 4000

              do SetOutputFilename

  EMBED: %ProcedureRoutines 500

    SetOutputFilename   ROUTINE
        if clip(txaname) = ''
            outputfile = ''
        else
            OutputFile = clip(TxaName) & '.embeds.txt'
        END
        display(?outputfile)

  EMBED: %DataSection 1300
```

```
parser          &TxaParserClass
q              TxaProcedureQueue

EMBED: %ControlEventHandling ?go Accepted 2500

    parser &= new TxaParserClass()
    setcursor(cursor:wait)
    parser.SetQueue(q)
    parser.Reset()
    if parser.parse(TxaName)
       if not parser.Write(OutputFile)
          message('Unable to write output file: ' & parser.getlasterror())
       END
    ELSE
       message('Unable to parse txa: ' & parser.getlasterror())
    END
    do Viewer4:Initialize
    display()
    setcursor()
    dispose(parser)
```

There's still a bit of embed code there, but now none of it contains any significant business logic. All the critical bits are inside my parser class.

There *is* a template that does this already, you know

Even back when I wrote the original parser I was pretty sure there was a template that would extract embed points, and

Steve Parker finally pointed it out to me: Bo Schmitz' free BoTpl utility can extract embed points from applications. Bo's excellent template does this by exporting a TXA and parsing the result. Sound familiar?

As useful as Bo's template is (and I highly recommend you get it and use it) I think it also points out the value of putting business logic into classes rather than into templates. Source code can be easily tested, and even debugged with the rudimentary Clarion debugger; templates are much more difficult to test and there is no true template debugger, only logging tools.

No, I'm not done yet

As has often happened to me, in the course of writing my tests I discovered some new features I wanted to implement, and also a bug or two that my initial set of tests hadn't uncovered. I'll explore these issues next time; after that I'll detour into an explanation of the CTest test runner app, and then it'll be on to the semi-real-world example of the Invoice app.

The downloadable source is a C7 application containing four APPs and a source code project. You can also load up any of these projects individually.

- CTest - the test runner application
- Embeds - the original embed application from the article series (doesn't extract embed source)
- ListEmbeds - the new app to list embed source
- TxaParser - a hand coded project containing the TXA parser class and supporting code
- TxaParserTest - an app containing unit test procedures for the TXA parser

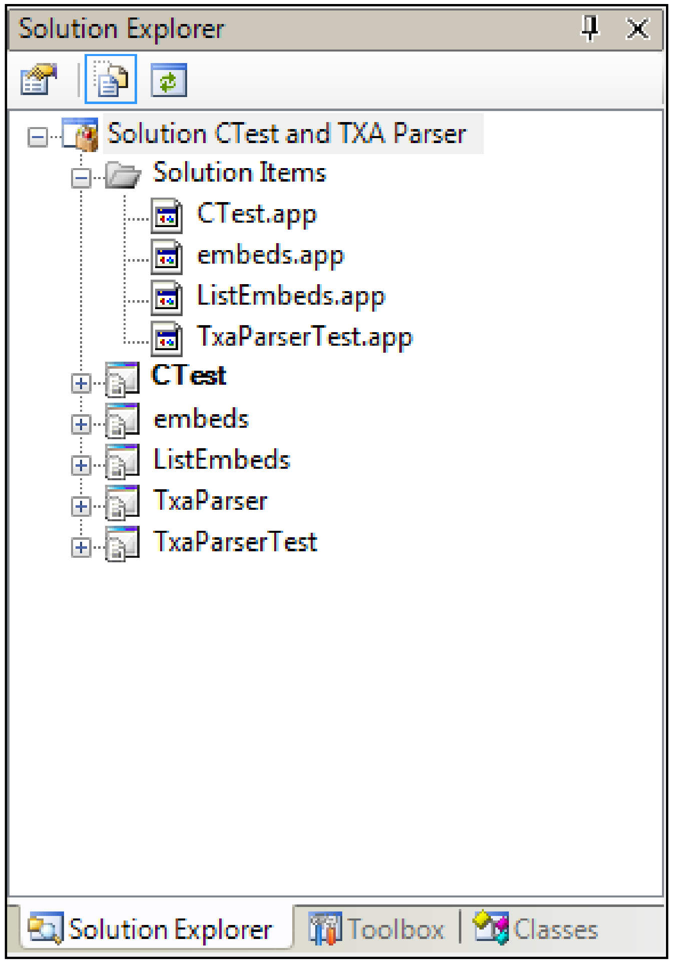


Figure 3. The solution

As well there are libsrc and template directories containing supporting templates and classes you may want to copy to your libsrc and template directories.

Download the source with DLLs (15 megs)

Download the source only (1 meg)

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

Reader Comments

Add a comment