Clarion Magazine

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

Clarion News

Read 20 Clarion news items.

The ClarionMag Blog

Read 5 blog entries.

Articles

Possible ClarionLive! DevCon In October

July 2 2010

Arnold Young and John Hickey are proposing a ClarionLive! DevCon in Denver CO at the end of October, to be held in conjunction with Capesoft's WebShop NetTalk seminar.

Preloading Queries With Clarion's QBE Template

July 16 2010

Query By Example (QBE) is a popular way to give users added flexibility when querying data. There are many advantages to using third party QBE templates, but Clarion's own QBE template is also quite capable. It's even better when you can preload your application with queries that show the users how to use QBE. Rip Peterman explains.

Colorado DevCon Gets Green Light

July 16 2010

John Hickey and Arnold Young have had a solid survey response and are moving ahead with plans for a ClarionLive! DevCon at the end of October, in Denver Colorado.

When Not To Generate Code

July 20 2010

There's an odd setting in Application Options that tells the IDE to edit source errors from code embeds in the generated source, rather than inside the app itself. That seems like a useless option to have, but in fact it's qutie valuable, as Dave Harms explains.

An Important Change To ClarionMag's Free Article Policy

July 26 2010

Eleven years ago Clarion Online ceased publication, and subsequently transferred its content to Clarion Magazine on the condition that the articles be publicly available for two years. That policy is finally changing; next week the COL articles will become subscriber-only.

A .NET AppGen - From Microsoft

August 4 2010

On August 23 Microsoft will release a beta version of LightSwitch, an application generator for .NET. Dave Harms looks at the LightSwitch announcements and wonders about the Clarion.NET AppGen and SoftVelocity's lack of communication.

C7 Conversion Tips

August 5 2010

Abe Jimenez provides some tips on converting your C6 applications to C7 (free access, not a ClarionMag article)

Roadmap updated, ClarionMag office closed August 9,10

August 7 2010

The Clarion Magazine office will be closed through Tuesday, August 10. We'll be back on Wednesday. Meanwhile, take a look at our updated roadmap to Clarion development! It's not quite done yet (still a few hundred articles to slot into place, and some corrections to be made), but it'll give you a good idea of the scope of ClarionMag's content. And be sure to read the instructions on how to create your own customized view of the roadmap!

Custom Queue Sorting, Part 1

August 9 2010

Clarion 5 introduced the ability to sort queues with a function call. But why would you want to do that? Simon Kemp explains not just why, but how, with a class and an easy to use template. Part 1 of 3.

Custom Queue Sorting, Part 2

August 12 2010

Clarion 5 introduced the ability to sort queues with a function call. In this second of three parts Simon Kemp goes over his sorting class and the accompanying template.

Custom Queue Sorting, Part 3

August 13 2010

Clarion 5 introduced the ability to sort queues with a function call. In this third installment Simon Kemp takes a closer look at the final version of the sorting function.

Are you getting the most out of ClarionMag? (screencast)

August 24 2010

We've added a bunch of new features to the Clarion Magazine web site. Are you getting the most out them? In this screencast Dave Harms covers the article lists, navigation, the new Clarion developer roadmap, and much more.

Microsoft LightSwitch: Impressive, but not a tool for building big apps

August 26 2010

Microsoft's LightSwitch product is now in public beta. It's is an impressive achievement, but at present, says Dave Harms, it's better suited to small .NET apps than to the kind of large-scale apps many Clarion developers create.

Clarion News

Icetips Previewer Build 2.7.349

Icetips Previewer build 2.7.349 is now available. There was a problem with the Previewer and CPCS reports that could cause invalid code to be generated for the CSV export class. This blog post also contains important information for Clarion 7 users dragging control templates to the toolbar.

Posted July 7 2010 (permanent link)

xWordCOM For C7.7248

The xWordCOM library has been compiled for Clarion 7 Build 7248.

Posted July 7 2010 (permanent link)

gCalc To Be Clarion 7 Compatible

Noyantis is in the process of fully overhauling the gCalc product and hope to have a new release available within the next week or so. The new version will be v7.00 and while being fully backwards compatible, it includes some major improvements to both the implementation and the visuals. The product now consists of the template and a new Calculator class - all source code. From v7.00 onwards, there will be no need to ship any extra DLLs etc with your application, all of the Calculator styles will be automatically compiled into your app. The price for a single developer license will be $79 new, $49 upgrade. These prices will include 12 months of updates. Multi developer licenses are available.

Posted July 7 2010 (permanent link)

BIT and BFET for C7.2

Comsoft7 has updated the installers for Big Image Tamer and My Favorite Embeds Template for C7.2.

Posted July 7 2010 (permanent link)

Comsoft7 and KwikSystems 241 Sale

Comsoft7 and KwikSYSTEMS are offering a 2-4-1 (two for the price of one) limited time sale until July 31, 2010 @ 11:59 PM Central Standard Time. (GMT -6 Daylight Savings Time). Purchase any product offered by Comsoft7 or KwikSYSTEMS Associates and get

any additional product of equal or lesser value as a bonus. Advertised Bundles are considered as a single product for these purposes.

Posted July 7 2010 (permanent link)

vuMail 3.18

vuMail version 3.18 has been released. This is a maintenance release that fixes a problem with embedded images not displaying properly in Thunderbird and other non-MS email clients.

Posted July 19 2010 (permanent link)

xButton C7 Compatible

An updated version of xButton with Clarion 7 compatibility is now available.

Posted July 19 2010 (permanent link)

FinalStep 2.25 / PseudoMax 1.03

Changes in FinalStep 2.25 / PseudoMax 1.03 include: PseudoMax, in Global Properties you can find now a new option "Force IMM property (compatibility with ERS)" for proper behavior with the EasyResizeAndSplit template; FinalStep on Clarion 7 (Legacy), fixed an issue that was generating a variable of the same template; PseudoMax on Clarion 7, fixed an error when trying to enter the local extension.

Posted July 19 2010 (permanent link)

FullRecord 2.25

Changes in FullRecord 2.25 include: Local thread attribute removed, detected by Clarion 7; IMDD sample was installing the wrong DCT.

Posted July 19 2010 (permanent link)

Clarion List & Label Interface Now Open Source (Updated)

Russ Eggen has placed the sources for the interface to List & Label on Source Forge, now using the permissive BSD license.

Posted July 19 2010 (permanent link)

xPictureBrowse, xDataBackup Manager Pro Updated

xPictureBrowse has been updated for Clarion 7, and a small correction has been made to xDataBackup Manager Pro related to Clarion 7 (incorrect display the window).

Posted July 31 2010 (permanent link)

Data Conversion Template 1.87

Data Conversion template version 1.87 has been released. Changes include: Improved C7 support; Fixed issue when using SYSTEM{PROP:DataPath}.

Posted July 31 2010 (permanent link)

Clarion2Java Updated

This release represents number additions to further support goal of clarion2java to be able to compile and run any product grade clarion project. A few key things: Finalized support for multibyte character sets. Multibyte strings now work everywhere, in STRING and other variable primatives, on screen, reports and on database storage. Adding multibyte chars like Chinese/Thai is as simple as just typing them into your source code. No special charset conversion or other programmatic activity required. Finalized how ARRAYS and Reference variables which have THREAD/OVER modifiers set on them compile into java. Drag/Drop support. Tweaks to compiler behaviour where clarion2java compiler did not behave exactly same as Clarion. Many other tweaks and additions. Complete changelog is in the download on sourceforge, in a file called CHANGELOG

Posted August 11 2010 (permanent link)

StrategyOnline Site Redesigned

StrategyOnline has launched a new, revamped version of its web site. The new site is easier to navigate, and features a range of new features including better tutorials, FAQs, product documentation, a newsroom, and more.

Posted August 11 2010 (permanent link)

Clarion Desktop 4.50

Clarion Desktop 4.50 is now integrated into the C7 IDE.

Posted August 11 2010 (permanent link)

Clarion Accessory Documentation

StrategyOnline has released new Clarion Accessory documentation totaling almost 500 pages, and available in three formats.

Posted August 11 2010 (permanent link)

Twelve Clarion Accessory Releases

StrategyOnline has released twelve Clarion Accessory updates, including J-Html 2.60 and J-Depends 1.03.

Posted August 11 2010 (permanent link)

FileTuner 0.52.

New in FileTuner 0.52: Legacy template is now separated from ABC template.

Posted August 11 2010 (permanent link)

Clarion Accessories 50% Off

To celebrate the launch of the new website, StrategyOnline is offering a 50% discount on all of its Clarion Accessories during the month of August. This applies to purchases made

through the StrategyOnline site only.

Posted August 11 2010 (permanent link)

Noyantis G-Calc 7.00

Noyantis has released version 7.00 of the G-Calc template. This release sees a full overhaul of the entire product, from raw TPL to finished user interface. The new version is fully backwards compatible, but includes some major improvements to both the implementation and the visuals. The product now consists of a template and a new Calculator class - all source code. From v7.00 onwards, there will be no need to ship any extra DLLs or LIBs etc with your application, all of the Calculator styles and required images will be automatically compiled into your app. Changes include: 12 different calculator styles including Windows Vista and Windows 7; Equation Solver (type in an equation and G-Calc will calculate the answer); Running tape facility (Display and print your calculations); Full Source Code included - No DLL or LIB required; Multi-paste options; Hot keys; Auto-load of field content (if numeric); Load and Paste to field with focus; c55, c6.x, c7.x, ABC, Legacy, Single EXE and Multi DLL compatible. The Equation Solver part of the template is currently being finalized and tested so will be included in v7.01. Enhancements over the previous versions of G-Calc include: Implementation of BIDMAS / BODMAS rules on all calculations; Implementation of Calculator Class; Removal of Calculator LIB / DLL requirement; A new 'Auto Open' mode added to Tape function; COS, SIN, TAN and MOD functions added; Windows Vista style added (standard + scientific); Windows 7 style added (standard + scientific); Length of calculated value increased. The price for a new, single developer license is $79; upgrades from any previous version of gCalc are $49. These prices include 12 months of updates. Multi developer licenses are available.

Posted August 11 2010 (permanent link)

The ClarionMag Blog

New T4 blog post by SV

There's a new post on the SoftVelocity blog that talks about the T4-based template language.

Posted July 7 2010 (permanent link)

Home page fix

For the past few days the home page was showing a "your subscription is up for renewal" message if you weren't logged in. This has been corrected - my apologies for any confusion that may have caused.

Posted July 22 2010 (permanent link)

Special RTL to track down missing menu bug

SV has released a special RTL build for C7.2 to help track down the missing menu bug. From a newsgroup post by Bob Z:

> There are several reports about Menu items missing at Runtime. Restarting the program restores the missing items.
>
> Attached to PTSS report 36756 is a special variant of the C7 RTL that outputs debug information about the Menu and its Menu items to help isolate this problem.
>
> If you have seen this problem in your program you can help us isolate and fix the problem with the following steps:
>
> - unzip the attached C70RUN.dll into your APP folder (_NOT_ your IDE BIN folder)
> - start DbgView.exe (download DbgView.exe from http://technet.microsoft.com/en-us/sysinternals/bb896647.aspx)
>
> - run your APP
> - if you see Menu items are missing, exit the APP
> - switch to DgView and do File->Save and then post an attachment to PTSS 36756 to this report with the DbgView log file
> - when you post the DbgView log, remember to specify which Menu items were missing
> - to post an attachment to the PTSS report open the PTSS report and press the hyperlink " Add additional information on this report..."
> - in addition, in the attachment (if possible) please include the program your were running. In the very least please specify what Menu Style (if any) that you were using, a copy of the WINDOW structure containing the Menu, and specify the OS you saw the problem on.
>
> Thanks, and I hope this helps to quickly resolve this rare but very important bug.
>
> -
> Robert

Posted July 28 2010 (permanent link)

Bob Foreman on ClarionLive

Bob Foreman will be presenting this Friday's ClarionLive! webinar. Here's Bob's promo from the ClarionLive! home page:

> *Best Development Practices in Clarion 7.2*
>
> *In the past few months, we have had many long time Clarion users step up to Clarion 7.2, and on behalf of SoftVelocity we thank you for your continued support.*
>
> *Many of these new users of Clarion 7.2 have been using Clarion for quite a while, and have a good familiarity with the Clarion*

Language and applications, but the new IDE of Clarion 7 is a major departure with regards to its flexibility and design flow.

In this webinar, I will review my best practices for application design. We'll review the application migration process and explore the many new areas of the IDE.

This session will include:

-Setting up your environment for a smooth development experience, tailored to your styles and preferences.
-General Dictionary and Application Best Design Practices and Work Flow.

-Review of the latest enhanchements to Clarion 7.2, including:
Extended Visual Styles support
Expanded Smart Formatting Options
Enhanced project support for copying multiple files
Other miscellaneous features.

As always, a Q&A will follow the presentation.

Posted August 18 2010 (permanent link)

ClarionLive DevCon Rooms 80% Booked

John and Arnold are reporting that the block of rooms reserved for CLDC attendees at the end of October is now 80% taken, so if you want to be sure of getting the excellent conference room rate book now.

Posted August 27 2010 (permanent link)

Possible ClarionLive! DevCon In October

Posted July 2 2010

ClarionLive's Arnold Young and John Hickey are considering putting on a ClarionLive! DevCon in Denver, Colorado in October 2010. This event would be in conjunction with (and following) a Capesoft WebShop training seminar. Although this proposed conference will be on-site in Denver, it will also have an online component for both the DevCon and the WebShop. You can attend in person or online.

These are tentative plans, and Arnold and John will be putting up a survey soon (perhaps Monday, July 5).

The proposed schedule includes:

- Capesoft World-Wide-WebShop (NetTalk seminar), October 27-28 (Wed-Thu)
- ClarionLive! DevCon, October 29-31 (Fri-Sun)

Tentative pricing:

- WebShop (NetTalk) two days: $550, or $400 if you register before July 30.
- ClarionLive three days: $550, or $400 if you register before July 30.
- All five days: $850, or $600 if you register before July 30.

Check the ClarionLive web site next week for more information.

Bruce Johnson has said that the Denver WebShop event will go ahead in any case, although the dates may change slightly if there is no DevCon.

Article comments

by Hyrum Tatton on July 7 2010 (comment link)

Please do both. I will be there. Hyrum

BACK TO TOP

Preloading Queries With Clarion's QBE Template

By Rip Peterman

Posted July 16 2010

There are many advantages to using third party templates for doing complex queries, but for those browses where limiting the number of records being displayed involves only a few criteria it is well worth looking at the Clarion templates, and in particular the Query By Example (QBE) template.

My problem: In a particular TPS file, only a few data fields would be of interest to the user for performing a search, yet the data could be well over 100,000 records. Even though I have keys to sort the records in orders that do aid the user in finding a record, or group of records, there are many other possibilities of interest whereby creating more and more keys is not the answer. I want to filter the data of a sorted order such that only the records of interest are displayed.

Another problem is that of the tool to perform the QBE filtering. I have found that when my third party tool writes selected record locators to a tag file, there can be slow responses, especially on a complete refresh of the tag file after a large number of records were tagged.

I also wanted to let the users store their queries, as well as provide some pre-packaged queries. Therefore, I looked at applying the ABC QBE template for generating my filtering versus me doing a lot of hand coding (even if I felt there were only six permutations of queries).

My situation uses a TPS file storing the occurrences when a person has signed in to a health and fitness club. The first display criterion of interest to my users is to limit the records to a single day. Next, a further limitation on the time span within a particular day can narrow in on attendance during time sensitive events (including theft, breakage, and abusive language). Various other criteria could be applied, but these will suffice to demonstrate the implementation of the QBE template for searches that have a little complexity, yet would otherwise involve considerable hand coding for even these few criteria.

It is far easier to have users get up-to-speed on new tools like this if there is a way to fast track the user's learning the use of the new feature *within* the application. (We all know how well the users immediately bury themselves in the accompanying 'Updates to the User Manual' documentation – really, not so!)

The QBE template has a great feature – a Save and Load capability using sections in the application's INI file. Now, if I can develop a method of creating some stored queries for the first time that a user makes use of the new QBE feature, that would aid in my ambition considerably. Also, in clicking on the Query Tab of the QBE list window, the user will see the composition of the query, itself. *Two birds with one stone* – fast-tracking to use the QBE and quick training on query construction.

As will be seen in the operation of the QBE button, if there are any stored queries, the QBE template will give the user an option to load one from a displayed list. This is accomplished by putting a check against the 'Use on startup' option of the QBE control (as shown later).

The problem I ran into was getting the query created in the INI and having the listing of these queries appear upon first entry into the Browse. I persevered, and will describe the solution in this article.

Getting started

Let's get started by selecting the QBE control template (Figure 1).

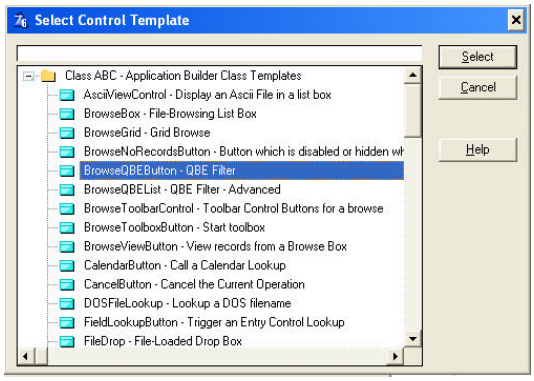


Figure 1. Selecting the QBE control template.

I've included an example app based on the People app. Figure 2, however, is a screen shot from one of my commercial apps. I've placed the Query button on the Browse Club Usage Records window and the `QueryStrg` variable (for information purposes) is at the bottom of the window.

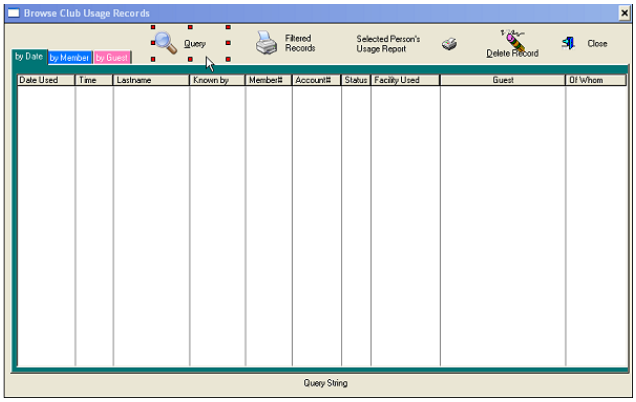


Figure 2. Query Button on Browse

Clarion provides QBE in two formats: list and form. By experimentation, I developed a preference for the List format over the Form. Here is the over-riding decision maker for me.

To perform a compound search on a data item in the Form, you need to load the control more than once with those fields for which this action would occur. Case in point: for a date range, the field would have to be loaded twice. This is show in Figure 3, second and third lines.

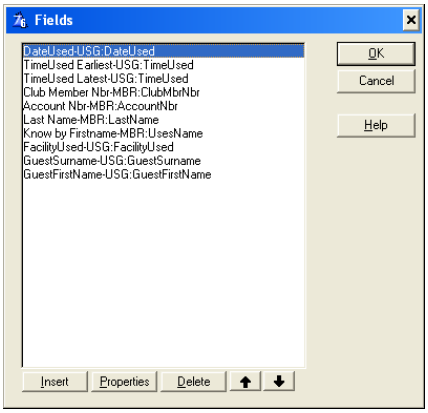


Figure 3. Query Fields Specification using Form

This specification would then produce the Query Form with the two Time Criteria and presented to the user as per Figure 4. There is no way to specify a compound search within the single data field.

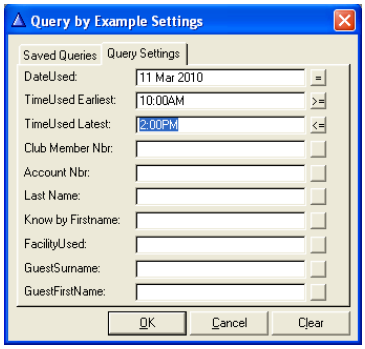


Figure 4. Query Fields for Search Criteria using Form

In specifying the fields for a List format, I only have to consider the ones upon which searching should occur. No duplications, just a clean list (Figure 5).

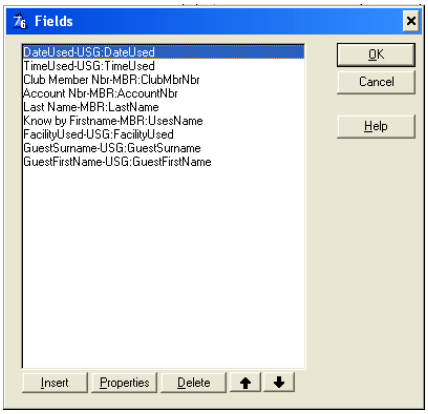


Figure 5. Query Fields Specification using List

When using the List format, the user can ‘Insert’ additional lines for a particular data item at run time (Figure 6) using the button at the bottom of the screen. I, the programmer, do not have to think of all possible compound searches nor define the search for the greatest possible compounding in order to cover all possibilities. That just makes for an overly

cluttered screen.

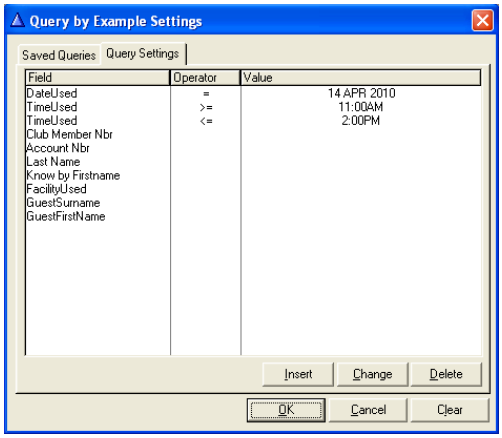


Figure 6. Query Fields for Search Criteria using List

One other compelling aspect was that of Save and Load of a query. When using the Form for the compound data field search criteria, the Save worked but the Load did not. It would only return the last value of the duplicate data items. (This may be corrected in Clarion 7.)

Here is the specification of the QBE Control's Options (Figure 7) to have it perform as I have found to be most advantageous.

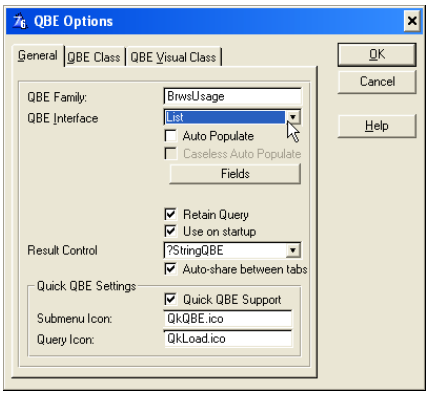


Figure 7. QBE Control Options

Because I want to tailor the control to my users' needs (as I know them to be),

I have picked the QBE Interface to be 'List' as evaluated above to be more suitable.

The 'Retain Query' checkbox redisplays the previous criteria each time you return to the query's composition screen, versus clearing the query criteria. I cannot think of why not to use this as the retained query can always be rejected by clicking on the 'Clear' button.

> **IMPORTANT**: The 'Use on startup' checkbox will open the QBE List to the 'Saved Queries' tab. This is exactly what I deem most beneficial to my users in repeating queries and learning how to construct a query from some existing ones.

I have tabs on my Browse, so I check the 'Auto-share between tabs' option.

I accept the default checkbox 'Quick QBE Support' with its data fields loaded. Here is the explanation from Clarion Help:

> Check the Quick QBE Support to enable special popup menu support for saved

queries. A View menu item will appear in the Browse box popup menu, with a sub-menu of saved queries. You can customize the View menu popup icon by modifying the Submenu Icon prompt. Customize the saved query items by modifying the Query Icon prompt.

I now specify the fields of the file to be searched. There are data fields I do *not* want the user to query, e.g. `PrimaryRecordNbr`, and `Updated`.

I have `DateUsed` and `TimeUsed` fields to satisfy my original requirement. I also give each field its own title (vs. the data element label) and a format or picture (whatever is easiest for user entry). I then included a few other fields, just to allow the user some freedom to do their own 'thing'.

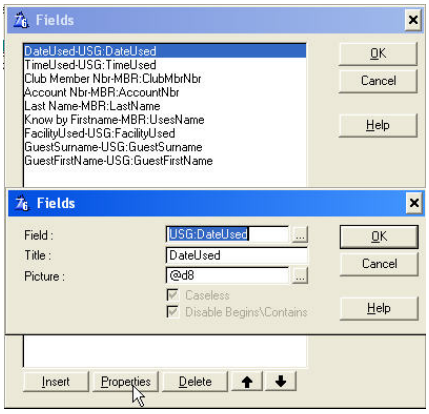


Figure 8. QBE Fields and Characteristics

I am done with respect to implementation of the QBE.

Prepared queries

Now, I want to help the user as much as possible. There are two occasions of help: 1) the first time the Query is used after the feature is implemented in the software and 2) each time thereafter. In the first case, because the checkbox for 'Use on startup' was selected, I need some initial queries to be loaded into the INI file, the first time, so that they can be listed and loaded. In the second case, where I can prefill some query data (such as a date that should be updated to today's date), then I would like to do this for my users, too.

The QBE Template stores information within the INI file as follows:

```
[__Dont_Touch_Me__]
Sectors=0
Sectors_1=BrwsUsage,StaffAdmit,Query
Sectors_2=BrwsUsage,TimeRange,Query
Sectors_3=BrwsUsage,TodayOnly,Query
Sectors_4=BrwsUsage,tsMRU,Query

[Query]
BrwsUsage$tsMRU$$Query=9
BrwsUsage$tsMRU$$Query_1=UPPER(USG:DATEUSED),,,
BrwsUsage$tsMRU$$Query_2=UPPER(USG:TIMEUSED),,,
BrwsUsage$tsMRU$$Query_3=UPPER(MBR:CLUBMBRNBR),,,
BrwsUsage$tsMRU$$Query_4=UPPER(MBR:ACCOUNTNBR),,,
```

```
BrwsUsage$tsMRU$$Query_5=MBR:LASTNAME,,,
BrwsUsage$tsMRU$$Query_6=MBR:USESNAME,,,
BrwsUsage$tsMRU$$Query_7=UPPER(USG:FACILITYUSED),,,
BrwsUsage$tsMRU$$Query_8=USG:GUESTSURNAME,,,
BrwsUsage$tsMRU$$Query_9=USG:GUESTFIRSTNAME,,,
BrwsUsage$TodayOnly$$Query=9
BrwsUsage$TodayOnly$$Query_1=UPPER(USG:DATEUSED),,>=29 MAR 2010,
BrwsUsage$TodayOnly$$Query_2=UPPER(USG:TIMEUSED),,,
BrwsUsage$TodayOnly$$Query_3=UPPER(MBR:CLUBMBRNBR),,,
BrwsUsage$TodayOnly$$Query_4=UPPER(MBR:ACCOUNTNBR),,,
BrwsUsage$TodayOnly$$Query_5=MBR:LASTNAME,,,
BrwsUsage$TodayOnly$$Query_6=MBR:USESNAME,,,
BrwsUsage$TodayOnly$$Query_7=UPPER(USG:FACILITYUSED),,,
BrwsUsage$TodayOnly$$Query_8=USG:GUESTSURNAME,,,
BrwsUsage$TodayOnly$$Query_9=USG:GUESTFIRSTNAME,,,
BrwsUsage$TimeRange$$Query=9
BrwsUsage$TimeRange$$Query_1=UPPER(USG:DATEUSED),=28 MAR 2010,,
BrwsUsage$TimeRange$$Query_2=UPPER(USG:TIMEUSED),,>=10:00AM,<=2:00pm
BrwsUsage$TimeRange$$Query_3=UPPER(MBR:CLUBMBRNBR),,,
BrwsUsage$TimeRange$$Query_4=UPPER(MBR:ACCOUNTNBR),,,
BrwsUsage$TimeRange$$Query_5=MBR:LASTNAME,,,
BrwsUsage$TimeRange$$Query_6=MBR:USESNAME,,,
BrwsUsage$TimeRange$$Query_7=UPPER(USG:FACILITYUSED),,,
BrwsUsage$TimeRange$$Query_8=USG:GUESTSURNAME,,,
BrwsUsage$TimeRange$$Query_9=USG:GUESTFIRSTNAME,,,
```

If I want to create an initial set of queries, then I am going to have to write records into the INI according to this structure. I can't just ship a new INI file because this is a software upgrade, and the users already have information stored in their INI files.

The easiest way to get these lines of code is to test the implementation of the Query button in the browse. It will write these records into the INI. Then all I have to do is cut and paste from the INI into a suitable embed point in the browse procedure, and wrap each line with an `INIMgr.Update` statement.

I start by loading the `FilterStrg` with today's date as stored in my global variable.

I then test for entries in the INI file. If none exist, then load the full query. If values exist, for a situation where there's a date value in the query, I update with today's date as set in FilterStrg.

After the [Priority 6500] in the Initialization of the Procedure is where I put the code.

```
CLEAR(GlobalRequest)              ! Clear GlobalRequest after storing locally
CLEAR(GlobalResponse)
! [Priority 6500]
FilterStrg='UPPER(USG:DATEUSED),,>='&format(GLO:CurrentDate,@D8)&','
IF ~INIMgr.TryFetch('Query','BrwsUsage$TimeRange$$Query')
  IF ~INIMgr.TryFetch('__Dont_Touch_Me__','Sectors_1')
    !want 1st to be tsMRU system default
    INIMgr.Update('__Dont_Touch_Me__','Sectors_1','BrwsUsage,tsMRU,Query')
    INIMgr.Update('Query','BrwsUsage$tsMRU$$Query',9)
    INIMgr.Update('Query','BrwsUsage$tsMRU$$Query_1',FilterStg)
```

```
    INIMgr.Update('Query','BrwsUsage$tsMRU$$Query_2','UPPER(USG:TIMEUSED),,,')
    INIMgr.Update('Query','BrwsUsage$tsMRU$$Query_3','UPPER(MBR:CLUBMBRNBR),,,')
    INIMgr.Update('Query','BrwsUsage$tsMRU$$Query_4','UPPER(MBR:ACCOUNTNBR),,,')
    INIMgr.Update('Query','BrwsUsage$tsMRU$$Query_5','MBR:LASTNAME,,,')
    INIMgr.Update('Query','BrwsUsage$tsMRU$$Query_6','MBR:USESNAME,,,')
    INIMgr.Update('Query','BrwsUsage$tsMRU$$Query_7','UPPER(USG:FACILITYUSED),,,
    INIMgr.Update('Query','BrwsUsage$tsMRU$$Query_8','USG:GUESTSURNAME,,,')
    INIMgr.Update('Query','BrwsUsage$tsMRU$$Query_9','USG:GUESTFIRSTNAME,,,')
  END
  INIMgr.Update('__Dont_Touch_Me__','Sectors_2','BrwsUsage,TodayOnly,Query')
  INIMgr.Update('Query','BrwsUsage$TodayOnly$$Query',9)
  INIMgr.Update('Query','BrwsUsage$TodayOnly$$Query_1',FilterStg)
  INIMgr.Update('Query','BrwsUsage$TodayOnly$$Query_2','UPPER(USG:TIMEUSED),,,')
  INIMgr.Update('Query','BrwsUsage$TodayOnly$$Query_3', |
     'UPPER(MBR:CLUBMBRNBR),,,')
  INIMgr.Update('Query','BrwsUsage$TodayOnly$$Query_4', |
    'UPPER(MBR:ACCOUNTNBR),,,')
  INIMgr.Update('Query','BrwsUsage$TodayOnly$$Query_5','MBR:LASTNAME,,,')
  INIMgr.Update('Query','BrwsUsage$TodayOnly$$Query_6','MBR:USESNAME,,,')
  INIMgr.Update('Query','BrwsUsage$TodayOnly$$Query_7', |
    'UPPER(USG:FACILITYUSED),,,')
  INIMgr.Update('Query','BrwsUsage$TodayOnly$$Query_8','USG:GUESTSURNAME,,,')
  INIMgr.Update('Query','BrwsUsage$TodayOnly$$Query_9','USG:GUESTFIRSTNAME,,,')
ELSE
  !this will update the date in the INI for each time the Query is used thereaft
  INIMgr.Update('Query','BrwsUsage$TodayOnly$$Query_1',FilterStrg)
END
FilterStrg=''
```

I use the `FilterStrg` variable to display the query at the bottom of the Browse window. So, I help them initially, big time, and from then onward in a small but effective manner.

Query details

The following `FilterStrg` value means “DateUsed data field must be Greater Than or Equal to CurrentDate”

```
FilterStrg='UPPER(USG:DATEUSED),,>='&format(GLO:CurrentDate,@D8)&','
```

This next statement updates the record in the INI file:

```
INIMgr.Update('Query','BrwsUsage$TodayOnly$$Query_1',FilterStrg)
```

I use the >= evaluator just to show its use over the simple = evaluation, which is shown next. The >= also lets the user easily back date the query to show every record from the specified date to the present.

For my filter of a time span within a specific date, I code the following records for the INI create/update, following the above code.

NB. The date is set to yesterday (`GLO:CurrentDate-1`)

```
FilterStrg='UPPER(USG:DATEUSED),='&format(GLO:CurrentDate-1,@D8)&',,'
IF ~INIMgr.TryFetch('Query','BrwsUsage$TimeRange$$Query')
```

```
  INIMgr.Update('__Dont_Touch_Me__','Sectors_4','BrwsUsage,TimeRange,Query')
  INIMgr.Update('Query','BrwsUsage$TimeRange$$Query',9)
  INIMgr.Update('Query','BrwsUsage$TimeRange$$Query_1',FilterStg)
  INIMgr.Update('Query','BrwsUsage$TimeRange$$Query_2', |
    'UPPER(USG:TIMEUSED),,>=10:00AM,<=2:00PM')
  INIMgr.Update('Query','BrwsUsage$TimeRange$$Query_3', |
    'UPPER(MBR:CLUBMBRNBR),,,')
  INIMgr.Update('Query','BrwsUsage$TimeRange$$Query_4', |
    'UPPER(MBR:ACCOUNTNBR),,,')
  INIMgr.Update('Query','BrwsUsage$TimeRange$$Query_5','MBR:LASTNAME,,,')
  INIMgr.Update('Query','BrwsUsage$TimeRange$$Query_6','MBR:USESNAME,,,')
  INIMgr.Update('Query','BrwsUsage$TimeRange$$Query_7', |
    'UPPER(USG:FACILITYUSED),,,')
  INIMgr.Update('Query','BrwsUsage$TimeRange$$Query_8','USG:GUESTSURNAME,,,')
  INIMgr.Update('Query','BrwsUsage$TimeRange$$Query_9','USG:GUESTFIRSTNAME,,,')
ELSE
  !this will update the date in the INI
  INIMgr.Update('Query','BrwsUsage$TimeRange$$Query_1',FilterStrg)
END
FilterStrg=''
```

There it is. Place this code at Initialize the Procedure in order to test to see if the records exist in the INI for my defined searches. If not, then I will write them. If they do exist, then I do a date update only.

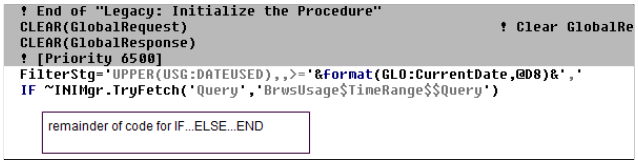


Figure 9. INI code insertion point

A little on these two statements as they may seem confusing:

```
IF  ~INIMgr.TryFetch('Query','BrwsUsage$TimeRange$$Query')
IF ~INIMgr.TryFetch('__Dont_Touch_Me__','Sectors_1')
```

This second IF statement on the first set of query creations is to ensure that the Most Recent Used (MRU) query (inherent in the QBE control operation) is stored as the first query group (not really necessary, but I like it there).

Here is my time range query.

```
INIMgr.Update('Query','BrwsUsage$TimeRange$$Query_2', |
  'UPPER(USG:TIMEUSED),,>=10:00AM,<=2:00PM')
```

That's it for the actual query statements.

Loading the queries

The next step is to ensure the Loading of the queries. I found that the queries I'd added to the INI file were not shown the first time I used the browse. I don't like telling users to get out of a procedure and come back in to make something work.

Still within 'ThisWindow.INIT', at the code insert after the BRW1.AskProcedure=1, I added these lines:

```
BRW1.Query.Restore('StaffAdmit')
BRW1.Query.Save('StaffAdmit')
BRW1.Query.Restore('TimeRange')
BRW1.Query.Save('TimeRange')
BRW1.Query.Restore('TodayOnly')
BRW1.Query.Save('TodayOnly')
```

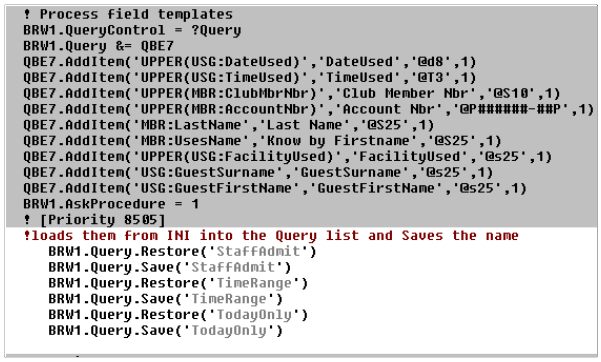


Figure 10. Restoring Saved Queries Insertion Point

This code solved the initial loading problem, especially since the saved queries do not exist on the first use of this upgraded procedure.

Printing the selected records

I mention this as my users seldom just want to look at records so, a little note on using the resultant queue for printing the filtered records is appropriate. I use a Global string variable (`GLO:Order`) for passing information from Browses to Reports. The report button is shown on the Browse window, right beside the Query button.

Insert the code in Figure 11 to pass the query string to the report.

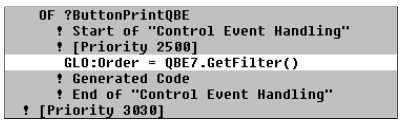


Figure 11. Assigning the QBE filter statement to a Global Variable

The **QBE7** reference can be seen in the source code at the queue definition or by the QBE Class tab on the QBE control.

In the Report Procedure, `Glo:Order` is tested for non-empty and used to filter the report, if exists.

In the section `ThisReport.Open` at the Priority 2500, put this code:

```
IF GLO:Order
    ThisReport.SetFilter(GLO:Order)
    ThisReport.ApplyFilter()
    GLO:Order = ''  !Clear the variable once used
END
```

```
ThisReport.Open PROCEDURE

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

! End of "Process Method Data Section"
  CODE
  ! Start of "Process Method Executable Code Section"
  ! [Priority 2500]
    IF GLO:Order
        ThisReport.SetFilter(GLO:Order)
        ThisReport.ApplyFilter()
        GLO:Order = ''  !Clear the var
    END
  ! Parent Call
  PARENT.Open
  ! [Priority 7500]

  ! End of "Process Method Executable Code Section"
```

Figure 12. Testing the QBE filter statement in the Report Procedure

Summary

I have always liked to reduce the amount of data being displayed in my Browses to that of 'information'. To accomplish complex filtering of the data in order to obtain a resultant set of records, Third Party programs have made my programming life easy. However, for a simple query, do not overlook the easy of implementation and the ease of functionality of the Clarion Template – BrowseQBEList. With the little bit of tweaking I do to provide initial queries for the user to make use of, this add-on template works wonderfully.

Download the source

Article comments

BACK TO TOP

Colorado DevCon Gets Green Light

Posted July 16 2010

As previously announced, ClarionLive's Arnold Young and John Hickey are looking atputting on a ClarionLive! DevCon in Denver, Colorado at the end of October 2010.

As announced at this week's ClarionLive!, barring an alien invasion, CLDC is a go!

They think they've found a place. It has really good internet. Details to follow once everything's firmed up.

Early bird special will be extended to August 15th.

Price of the conference *may* change.

Figure 1 shows the preliminary schedule. The x indicates a session, so while there are three tracks planned, everyone's together for the keynote on Friday and the wrap-up on Sunday. Session blocks include either:

- One 90 minute session
- Two 45 minute sessions
- Three 30 minute sessions

Note the planned vendor rooms, open lab sessions and Sunday's application showcase.

Thursday October 28				
5:30p	Registration			
7:00p	Welcome Party			
	no clarionlive webinar this week	*Sessions are either 3-30 or 2-45 or 1-30/1-60 or 1-90 minute session w/ 10 minute Q&A*		
Friday October 29		**Session 1**	**Session 2**	**Session 3**
7am	Registration/Continental Breakfast	x		
8am	Keynote Block	x		
8:30am	Block 1	x	x	x
10am	Break			
10:30am	Block 2	x	x	x
12 noon	Lunch Break			
1:30am	Block 3	x	x	x
3pm	Break			
3:30p	Block 4	x	x	x
5:00p	Vendor Rooms	xx		
6:00p	Dinner Break			
7:30p	Optional Extra Open Lab Session??	xx		
Saturday October 30		**Session 1**	**Session 2**	**Session 3**
7am	Continental Breakfast	x		
7am	Application Showcase	x		
8:30am	Block 5	x	x	x
10am	Break			
10:30am	Block 6	x	x	x
12 noon	Lunch Break			
1:30am	Block 7	x	x	x
3pm	Break			
3:30p	Block 8	x	x	x
5:00p	Vendor Rooms	xx		
6:00p	Dinner Break - Presenter Party (?)			
7:30p	Optional Extra Open Lab Session??	xx		
Sunday October 31		**Session 1**	**Session 2**	**Session 3**
7am	Continental Breakfast	x		
7am	Application Showcase	x		
8:30am	Block 9	x	x	x
10am	Break			
10:30am	Summary/Give-aways Block	x		
12 noon	Finish/Farewells			

Figure 1. Tentative schedule

For more information see the original announcement here in ClarionMag, or check the ClarionLive! site.

Article comments

by Stuart Andrews on July 19 2010 (comment link)

Wowza! This is great news!

Wish I could be there. Am sure it's gonna be a cracker.

by Geoff Spillane on July 20 2010 (comment link)

I have spoken to John and Arnold a few times about this DevCon and I can tell you that their dedication to intense preparation will absolutely guarantee success. I recommend to everyone to try to get there if at all possible. You'll learn heaps and make some really great friends.

BACK TO TOP

When Not To Generate Code

By Dave Harms

Posted July 20 2010

A little while ago, deep in the bowels of a medium-sized office building somewhere on the sprawling ClarionMag campus, the following discussion took place:

> **DenizenA**: I work in an embed and have a syntax error. I compile and down below the procedure tree, I see my errors. I double click on one and pop up in the generated CLW for that procedure. That's not exactly optimal. Do I have some setting wrong? Shouldn't it bring me to the right embed point and let me edit the code there?
>
> **Me**: Go to Application Options, first tab, make sure the fourth-last item, "Edit embedded source errors in generated code" is *not* checked.
>
> **DenizenA**: Yes. 4th up from the bottom "Edit Embedded Source Errors in Generated Code" is checked. Unchecking it and testing ....
> Oh. How lovely <g>. Just what the doctor ordered. Thank you, Dave. Why wouldn't it be set that way already? I mean, who would edit generated source?
>
> **DenizenB**: I still don't know why that option was added - long before C7. BTW - you don't need to double click the error, single click works.

This discussion brings up two useful points. One, that you don't need to double-click the error (thanks DenizenB - I'm still unlearning that habit), and two, that there is in fact a reason you might want to check the "Edit Embedded Source Errors in Generated Code" option, and it has to do (mostly) with debugging.

Separating code from the AppGen

One of the very first pieces published in ClarionMag, way back in 1999, is titled The Clarion Advisor: Speed up your APP debugging with a PRJ. The premise of that article, which I encourage you to read for the overall concept if not the implementation, is that sometimes you need to hack away at your code to find the solution to a problem, but in doing so you really don't want to mess up your APP file. You can do this by extracting the project information, which is what Clarion needs in order to build your application from source, and

then you use this information to compile your code *as if it were a handcoded project*.

Getting at the project data is easier than it was back in '99, and easiest of all in Clarion 7, since the project data exists as a node in the Solution Explorer. An in fact you can choose to just build the source (without generating first) several ways: by pressing F8, by clicking on the Start without Debugger (Builds Solution) icon on the toolbar, or by selecting build for just the specific project as shown in Figure 1.

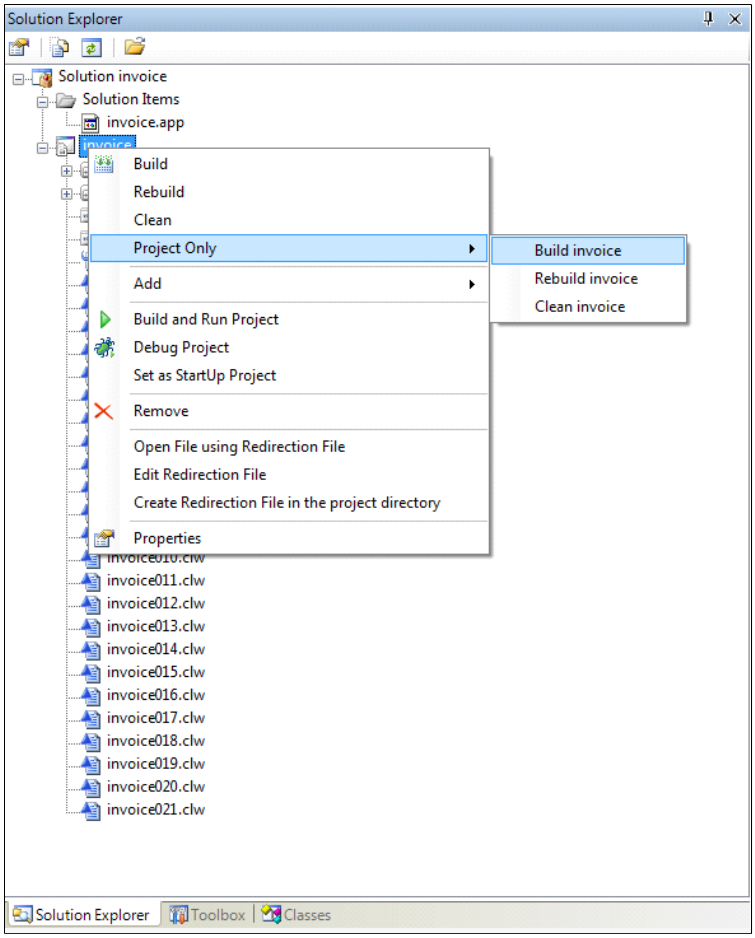


Figure 1. Building a project's source

So let's say you've got some crazy bug you're trying to track down, and perennial favorites STOP and MESSAGE and even the scintillating debugger can't shed any light. So you decide to start hacking and chopping away at your source code (per the article previously cited) in an attempt to isolate the problem. But as you hack and chop you introduce compile errors. And if any of those compile errors are of the sort the IDE can identify as being embedded code (I'm not sure how it does this) it will, by default load up the app. But that is exactly what you do *not* want because you're in exploration mode, and you don't want the AppGen recreating all the code you've just hacked up (at least not until you've isolated the cause of the problem).

And that's when you close any open apps, go to Tools | Application Options, and *check* the

"Edit embedded source errors in generated code" option. That way the IDE will take you straight to the source where you can continue to wreak havoc uninterrupted by freshly generated code.

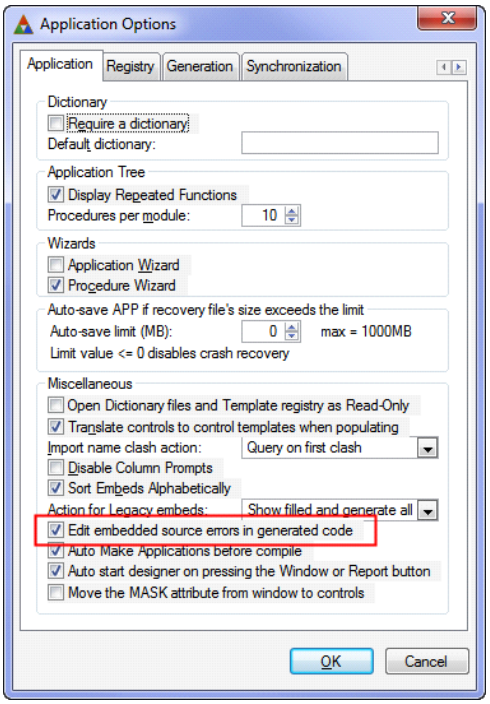


Figure 2. Application Options - editing embed errors in generated code

I can't think of any reason for using this option outside of debugging and experimentation, but it's very handy in those circumstances.

And make sure it's off, otherwise

If you participated in the early betas of C7 you'll know that initially the IDE always took you to source code, even when the error in question was in embedded code. Eventually that got fixed, but in DenizenA's case the option was on by default, as it was (if I recall correctly) in those early days. Reportedly a fresh install (no pre-existing IDE data) defaults this setting to false.

If you ever notice that clicking on an error takes you to generated source when it should be taking you to an embed in the app, check your Application Options.

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

Article comments

BACK TO TOP

An Important Change To ClarionMag's Free Article Policy

Posted July 26 2010

At midnight on Monday, July 26 2010 we'll be raising Clarion Magazine's subscription rates. Shortly thereafter we'll also be making an important change to our free article policy.

Back in 1999, when Clarion Magazine went online, it was one of two Clarion publications. The other was Clarion Online (COL), which began publication in 1997. COL went belly up in 1999, and I entered into an agreement with Tom Moseley, COL's publisher, to carry the COL articles on the ClarionMag site with the guarantee that those articles would remain freely available for a minimum of two years.

That was eleven years ago.

We're in the process of changing that policy. Sometime next week access to the Clarion Online articles will be restricted to Clarion Magazine subscribers. You won't need a current subscription to access those articles - any regular subscription, even expired, will do the job.

If you have any questions or concerns, please post a comment to this article or send me an email.

Article comments

BACK TO TOP

A .NET AppGen - From Microsoft

By Dave Harms

Posted August 4 2010

I remember Bruce Barrington, Clarion's founder, telling a story he'd heard of Bill Gates calling some of his key people into his office. Gates had a stack of Clarion Professional Developer boxes sitting on his desk. "Take these, and tell me why these guys can do something we can't do," he said (or words to that effect).

I've never been able to verify that story, and Bill doesn't sit at that desk anymore, but I do know that MS never did develop anything that approached CPD's ability to create usable, customizable applications without writing all that grunt code. Nor did Microsoft create a Clarion-like product for Windows (prior to .NET).

While .NET has made some aspects of writing business applications easier, the barrier to entry is still pretty high. You have to settle on a data layer technology, you have to create all of your windows manually, and you have to plumb the UI to the data layer and integrate all of your business logic. There are some partial code-generation solutions, mainly from third parties, but until now I haven't seen anything in .NET like Clarion's Win32 template-based system.

Until now.

On August 23, 2010 Microsoft will release an AppGen-like product called LightSwitch. This is a .NET development tool that's available as a separate Visual Studio-based product (which will be priced somewhere below VS Pro) or as an integrated tool within the full version of Visual Studio.

So what does LightSwitch do? In short, it can generate a working application from an existing database, or from tables you define. LS has a data dictionary-like component that lets you set up validations, data formats and other options. It has templates (although the extent of the template functionality isn't yet clear - they may be quite different from Clarion templates). It has window designers (although they're not drag and drop, that I can see, rather they're generated based on other settings). You can write your own code and integrate it with the generated code. You can also take the generated source and work with it purely as source code, if you wish.

LightSwitch's key features include:

- A choice of language - you can create your apps in either C# or VB.NET
- A data model based on Entity Framework (think of this as similar to, but more than, FileManager and RelationManager), with the ability to also use data from RIA services and SharePoint.
- Support for Azure (cloud) apps
- MS Office integration (export to Excel)
- The ability to create desktop or browser apps
- The ability to modify an application while it is running
- Two and three tier deployment options
- Application themes
- Customizable templates
- Paged browses

Deployment

In beta 1 you will be able to deploy applications in three ways:

- As a two tier, client server desktop application (much the way Clarion apps are currently deployed)
- As a three tier desktop application, requiring a server running IIS to permit access to data
- As a three tier web application, requiring a server running IIS to permit access to data

It looks like all three options use Silverlight for the user interface, but the first two run Silverlight outside of the browser (OOB), while the third runs inside the browser.

Databases

As mentioned above, LightSwitch uses Entity Framework as the data layer so you can work with any database that has an ADO.NET Entity Framework provider. That should mean that most mainstream SQL databases are supported. I don't see an easy way to use TPS files.

Required skills

Microsoft appears to be positioning LS as a tool that enables non-programmers to build database applications (which makes me think of the glory days of Clarion Professional Developer). They've emphasized that while you can add in all sorts of custom code, you don't *need* to write any code to create a working application. That's also true of Clarion, but almost all Clarion devs write at least some code. So what kind of skills would you need to write LS apps?

Assuming a stock LightSwitch app won't do the job, you'll need to learn C# and/or VB.NET, as those are the two languages supported in the first release. You'll probably be doing most

of your data manipulation using LINQ to Entities, so get to know LINQ. And to modify anything related to the user interface you'll need some XAML knowledge.

What you won't need, it seems, is to know how to design the overall application. Like Clarion, LightSwitch gives you an instant application architecture

Will it compete with Clarion.NET AppGen?

Is LightSwitch the equivalent of the hoped-for Clarion.NET AppGen? It's too soon to tell. For one thing, we haven't yet seen the Clarion.NET AppGen. For another, LightSwitch is just heading into beta and its feature set hasn't been finalized.

With a focus on non-programmers, it may be that LightSwitch is too lightweight, that it can't be customized to the degree required by many Clarion developers.

But in the absence of a Clarion.NET AppGen, LightSwitch at least looks promising as a way to get started developing business apps in .NET. It's the most Clarion-like of all the .NET codegen tools I've seen.

And that makes the lack of meaningful information about the Clarion.NET AppGen all the more critical. Last fall SoftVelocity predicted a .NET AppGen beta release in the first quarter of 2010, with a gold release also possible in the first quarter. In a blog post following the Aussie DevCon SV even looked forward to the next Florida DevCon:

> Looking down the road, we hope to have our next US Developer's Conference in 2010 soon after the Clarion.NET Gold release. Probably in Orlando in the second quarter if everything moves ahead as planned.

The beta release has yet to materialize, never mind gold. A DevCon in 2010 is, well, implausible. And there has been precious little information from SV on the status of the AppGen. A July 6 post on the SV blog, the most recent in the AppGen.NET category, doesn't say anything about the status of the AppGen but merely recycles some template information previously released at the Aussie DevCon.

Requests for information, posted in the SV newsgroups, for the most part go unanswered or are deleted.

Meanwhile, Microsoft is less than three weeks away from a beta release of a product that is, at the very least, a big step toward meeting the promise that was Clarion.NET.

A few LightSwitch links:

- LightSwitch home page (watch the videos)
- A Channel9 video on creating apps with LightSwitch
- Blog post by Jason Zander

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

American Society of Journalists and Authors (ASJA).

Article comments

by Lemuel Turner on August 5 2010 (comment link)

Very interesting. I don't think I'll be putting anymore money into Clarion.NET. SoftVelocity would need to come up with a robust Win32 to .NET conversion program to interest me in investing more money in Clarion.NET.

by Graham Dawson on August 6 2010 (comment link)

"...It may be that LightSwitch is too ligthweight..."

That's what they said about Visual Basic when it was first released. If LightSwitch takes off then it will be expanded and extended beyond all recognition. Given that it's firmly based on all the latest technologies there should be no limit to the possibilities.

SoftVelocity had better get a move on otherwise it'll be 'LightsOut' at the Clarion.NET office.

by Robert Barton on August 8 2010 (comment link)

Ye Gods! The perfect clarion sales talk - this will really light the blue touch paper for Clarion at last!

Oh dear - the products name has changed - possibly not a good idea!

But wait - it gets even worse it's not a Soft velocity product - it's by MS!

The game appears to be up - time waits for no man(or even person!).

BB must be laughing his socks off!

by Robert Wagner on August 8 2010 (comment link)

"Requests for information, posted in the SV newsgroups, for the most part go unanswered or are deleted."

This behavior is most troubling. Obscene or off topic rants can well be deleted, but it appears that any post to the newsgroups that could be construed as negative gets deleted. While it may mean that, at a minimum, RZ reads every posting, it also converts the newsgroups into a house PR organ. Further, it can defeat the purpose of a newsgroup, which is to disseminate help and information, and provide feedback.

by Djordje Radovanovic on August 9 2010 (comment link)

Please SV, read my message that you deleted from news groups on 12/22/2009. Was I right or not? Yes, I was right and I am sorry that I was right.

by Robert Hutchison on August 9 2010 (comment link)

Djordje

Don't leave it like that, tell us what you said mate !

---

by Djordje Radovanovic on August 9 2010 (comment link)

I said that Clarion# is future for SV and Clarion developers and to put all their efforts on this product and to forget Clarion 7. This product did not bring us ability to write better programs. It was just changing environment and some polishing but nothing revolutionary as Clarion# is.

---

by Michael Dettmer on August 10 2010 (comment link)

... I had a bad feeling about SV already a long time. What about Bob Zaunere? Still there or not? And what is he doin? How does SV earn money and with what? Can that really work like it is?

So I think we all can be glad that MS is taking over part of the concept. Either SV is taking the leadership in generating .NET sourcecode or they will be history sooner or later.

Sorry for that.

---

by Michael Goosen on August 11 2010 (comment link)

What attracted me the most to Clarion was the productivity edge. The self-contained, solution-oriented approach that got down to the business of business. What I have always hated is Clarion's closed education model, it's high barrier to entry ("... it's a secret -- no, you have to invest first, then we'll show you, but not before ..." -- as opposed to the competitor eg. Visual Studio Express model), and as a result, it's lack of market penetration, with all the things that hang off of that, like skills recognition, employment opportunities etc.

With this news, it may be finally possible to have the best of both worlds -- productivity and market share that's meaningful.

---

by Michael Goosen on August 11 2010 (comment link)

Reading my post -- I forgot to elaborate on the closed SV education model -- TURBO PASCAL (see History of Clarion) was such a resounding success and took the market by storm precisely because it made it easy for students to get in on the developer action at a low, low cost.

SoftVelocity should be giving as much of their their education / training materials away (mostly soft, so zero cost anyway) for free to as many takers as they can possibly find out there (and maybe even bribing a few non-Clarion hardliners to look at it)), than guard their skills IP, protecting the precious few with a vested interest in keeping the key knowledge base under wraps. If I was them, I would rather have 1% of the entire

pie, than 100% of 0.0000000001% of the pie. Which is how it currently looks, tragically.

Take a look at "4GL Languages" -- on Wikipedia -- Clarion is not even mentioned. Out of a huge list of languages. That's how well it is kept under wraps.

I think I am not alone in my feeling that most Clarion programmers out there have a constant, day-to-day battle trying to promote Clarion to the non-Clarion world -- with little or no support from SV by way of educational materials, or trial versions of their products, to support the cause. Brings tears to my eyes actually.

In my opinion, if SV take a fall, it will ultimately not come down to quality of product, but to the fact that this bright shining Clarion Star was kept so well hidden, was such a great educational secret, that it finally simply starved from lack of interest in the greater [non-Clarion (ie. the target SV market) community] and died...

And I am writing this as a very, very loyal from the old C2.108 days Clarion developer.

Who can see the writing on the Great Microsoft Wall beginning to form.

It's not too late. But who will take up the challenge, and change this [almost certain] dismal future of Clarion to one where it takes its rightful place?

That's the question.

---

by Dave Harms on August 12 2010 (comment link)

Check out the the Channel 9 video Visual Studio LightSwitch - Beyond the Basics. Lots of good information there on how to extend LS.

---

by Dave Harms on August 20 2010 (comment link)

I've been playing with LightSwitch for a couple of days now, and I have to say it's a bit of a disappointment. While an exciting product in many ways, it's not an application code generator in the way Clarion is. It's more like Clarion Personal Developer - it creates a closed, interpreted application rather than compiled source code.

---

by Kosta Vasic on August 20 2010 (comment link)

I’ve been learning C# and I’ve been to a Silverlight class and I’ve looked at LightSwitch. This stuff is still Microsoft and they don’t seem to have a clue. In the Silverlight class they said that LINQ will go away and so will RIA services. (A Microsoft insider was teaching the class and said the ADO group had more power and didn’t like LINQ) If SV can get it to work soon, Clairon.net is still in the running. This is what Clarion has done in the past and as done well. Let me use a forms paradigm and drag and drop controls and let Clarion write all of the XAML behind the scenes. In each control where I write code let Clarion handle all of the C# behind the scenes and put that in the correct C#.cs file. We write data centric applications. The Model doesn’t

need to be changed: Menu, Grid, Form, Report, Process (Stored Procedure). This works and works very well. Microsoft changes stuff just for the sake of change. C# is case sensitive. Really? I thought we figured that out twenty years ago. So most guys in the class spent a lot of time debugging upper, lower case problems. They did that because it would have the Java feel… SV needs to come with a version one that will do this basic stuff. It will still be far ahead of Microsoft. Then reduce the upgrade cycle to every quarter. Within a year Clarion.net will be the product of choice. AlphaSoft has a product that has a similar paradigm and I've been to two of their week long courses this year and the classes were sold out. This product is needed!

---

by Dave Harms on August 20 2010 (comment link)

Kosta,

A .NET AppGen is most desperately needed.

As for LINQ itself, it will be around for a long time to come - it's a core technology. LINQ to SQL, however, is no longer being developed, as far as I know. It has been replaced by Entity Framework and LINQ to Entities.

The ORM war is pretty much over - you can still use older technologies, but until we all start using object databases you can expect ORMs like Entity Framework to be the norm for new development. You need something to translate between an object model and a relational database, and it's a bit silly to keep reinventing the wheel.

Dave

BACK TO TOP

Log On

Please enter your username and your password. *(Note: If you registered on or after May 19, 2010 your email address is your username.)*

If you've forgotten your password or your username, click here.

Don't have a login? Register now! It only takes a moment, and it's free (although most articles do required a paid subscription).

Account Information

Username:

Password:

Remember me: é

If you check this option you will be automatically logged in the next time you come back to this site. Your Username and password will *not* be remembered. If you want the Username and Password fields filled in automatically the next time you log in, please use your browser's "Remember login" feature.

Log On

Clarion Roadmap

Try the roadmap (beta)

Talk To Us!

Make, view, and vote on suggestions here!

Search ClarionMag

Go

Advanced search

From the archives

Using Local Classes Instead Of Local Routines

8/11/2005 12:00:00 AM

Local classes are often touted as a replacement for procedure routines. But what are local classes, why do you need them, and what can you do with them? Nardus Swanevelder answers these questions, and shows how easy it is to use local classes.

Links

Download PDFs
Recent comments
Free articles
Privacy policy

Custom Queue Sorting, Part 1

By Simon Kemp

Posted August 9 2010

Breaking news! Clarion 5 introduced the ability to sort queues with a function call. Seems that means I can use my own code to sort queues. Well, so what? Why on earth would I want to do that?

One of the C5 "what's new" help topics must have mentioned this addition, but it sailed straight over my head. Subsequently (and despite reading about it in the online help many times) I remained blissfully ignorant to the potential. But eventually the light went on. All that time messing around with obscure queue structures and workarounds so that the faithful `SORT` I knew would give me the desired result, when an alternative had been staring me in the face. Oh well, better late than never.

This article attempts to explain how that minor revelation eventually led to a class and template to handle column sorting in queue-driven list controls. Along the way I hope it will spread the word about custom queue sorting.

But first, a word on standard queue sorting.

Standard queue sorting

Queues are one of the Clarion language's gems. They have been around from the start (CPD, when they were confusingly declared as a `TABLE`). A queue such as this:

```
MyQueue        QUEUE
SomeString       STRING(30)
SomeDate         DATE
SomeTime         TIME
                 END
```

can be maintained and accessed by specifying one or more column names and a sort direction:

```
ADD(MyQueue, +MyQueue.SomeString, -MyQueue.SomeDate)
PUT(MyQueue, +MyQueue.SomeString, -MyQueue.SomeDate)
GET(MyQueue, +MyQueue.SomeString, -MyQueue.SomeDate)
SORT(MyQueue, +MyQueue.SomeString, -MyQueue.SomeDate)
```

Those versions of `ADD` / `PUT` / etc. are usually enough for the job in hand, enabling large

amounts of data to be stored, sorted and accessed in memory. But what happens when more control of the sort order is required? A good example (and common requirement) is removing case-sensitivity from the sort. Assuming I want a case-insensitive sort on the string `MyQueue.SomeString`, there are two immediate possibilities:

1. Always upper (or lower) the data before adding it to the queue

```
MyQueue.SomeString = upper(MyQueue.SomeString)
ADD(MyQueue, +MyQueue.SomeString, -MyQueue.SomeDate)
```

2. Add another upper or lowercased field for the sort

If `MyQueue` is being displayed to the user, it may not be desirable to use approach 1. Equally, `MyQueue.SomeString` may be required in its original state at some point down the line. So I recommend you just add another column and use that instead:

```
MyQueue         QUEUE
SomeString        STRING(30)
SomeDate          DATE
SomeTime          TIME
SomeStringLC      LIKE(MyQueue.SomeString)
                END

...


  MyQueue.SomeStringLC = lower(MyQueue.SomeString)
  ADD(MyQueue, +MyQueue.SomeStringLC, -My.SomeDate)
```

Prior to Clarion 5 this was the only documented approach to case insensitive sorting, and in this simple example it remains effective. But if there are multiple columns involved it quickly becomes a tedious coding chore and more esoteric queue sorting requirements can not be implemented this way. Enter stage left (with fanfare): functional sorting!

Functional Queue sorting

Look `QUEUE` up in the online help and you will find these forms of the four relevant commands:

```
ADD(queue , function)
GET(queue , function)
PUT(queue , function)
SORT(queue , function)
```

along with this explanation of how 'function' should be protoyped and coded:

> *function* The label of the function containing two parameters of a *GROUP or named GROUP passed by address, and having a SIGNED return value. Both parameters must use the same parameter type, and cannot be omitted. The RAW, C

and PASCAL attributes are not permitted in the prototype declaration.

It can take a while (for me at least) to appreciate that an extremely powerful tool has just been described.

One important thing to understand is that when the help says "two parameters of a `*GROUP` or named `GROUP`" it really means a function will be passed two elements of the queue to compare. And all it asks of the function is to return 0 (they are equal), -1 (parameter 1 is less than parameter 2) or +1 (parameter 1 is greater than parameter 2). Suddenly the limitations of standard queue sorting have evaporated. But that may not be obvious yet.

Returning to the case-insensitive example, here is the simplest and oft-quoted approach:

```
MyQueue        QUEUE
SomeString       STRING(30)
               END

...

  SORT(MyQueue, MySortFunction)

...

MySortFunction  procedure(*group a,*group b) ! ,signed

    code
    if upper(a) = upper(b)
      return 0
    elsif upper(a) < upper(b)
      return –1
    else
      return 1
    end
```

This works but is of little use unless the queue contains only a single string field. No knowledge of the group structure is used by the function, which relies on Clarion's default behaviour of treating groups as strings when comparing `a` to `b`.

To be of any practical use a sort function will usually need to understand the passed groups. There are two ways to do this: pass a known structure, or examine the passed structure.

1. Pass a known structure (ideally accompanied by some sort-specific instructions)

If a sort function is specific to a single queue then this is the way to go - write a function intended to deal with a specific queue structure. The case-insensitive example again:

```
MyQueue        QUEUE
SomeString       STRING(30)
SomeDate         DATE
```

```
SomeTime          TIME
                END


CompareMyQueue  GROUP(MyQueue),TYPE
                END

...

  SORT(MyQueue, MySortFunction)

...

MySortFunction  procedure(*CompareMyQueue a,*CompareMyQueue b) ! ,signed

    code

    if upper(a.SomeString) = upper(b.SomeString)
      return 0
    elsif upper(a.SomeString) < upper(b.SomeString)
      return –1
    else
      return 1
    end
```

But what about my "sort-specific instructions"? You might want the user to click a checkbox indicating the queue should be sorted in reverse order? Well, unfortunately Clarion doesn't let you pass any further parameters to the sort function so you have a choice to make:

a. Write multiple sort functions and make a decision in code as to which should be called. This always works but lots of queues and lots of "sort instructions" could get tedious to code.
b. Stick with a single sort function per queue but have it reference external data to determine how it goes about its business - Sounds better? Of course it does. But there's a subtle problem with this approach that will be explained a little later.

2. Examine the passed structure

By "examine" I mean use Clarion's `WHAT()`, `WHO()` etc.

This is the approach to use when implementing a generic sort function. Again, I'll skip this for now because it's covered in the details of the class implementation later.

Take a moment to look at what Clarion is providing here.

The structure and content of a queue is under your control, but you may need to compromise efficiency of that structure when relying on the standard queue sorting features. Functional sorting provides a different line of attack, enabling you to keep the queue "as you

want it" and write some code to get the desired sorting. It also means that queues can be sorted in ways that are impossible with the longer-standing forms of SORT.

Here, for me, is an excellent example of where functional queue sorting scores:

The following queue needs to store strings that might be between 25 or many 1000's of characters in size. There will be *lots* of entries so I am using string references in an attempt to minimize the memory hit:

```
MyQueue        QUEUE
SomeString             &STRING
           END
```

But this queue also needs to be sorted at some point using SomeString. Standard queue sorting - SORT(MyQueue, +MyQueue.SomeString) - certainly does something in the sense that queue order will change. But Clarion is not looking at the content of the referenced strings, more likely it is sorting using a combination of the string reference's address and size. Enter our hero again, and this time the sort is case-sensitive:

```
CompareMyQueue  GROUP(MyQueue),TYPE
                END

...

  SORT(MyQueue, MySortFunction)

...

MySortFunction  procedure(*CompareMyQueue a,*CompareMyQueue b) ! ,signed

    code
    if a.SomeString = b.SomeString
      return 0
    elsif a.SomeString < b.SomeString
      return –1
    else
      return 1
    end
```

The prototyping and seven lines of code above should reveal the full potential of functional queue sorting. There may be a better approach to the problem described - sorting an unknown number of strings of unknown length – but it would definitely involve a lot more coding.

Managing the downside

Unfortunately not everything in the garden is rosy. I have encountered two issues with

functional sorts that are worth detailing here:

Problem #1 - Speed

Compared to standard queue sorting, functional sorts can appear very slow. But that comes down to the code *inside* the sort function. The good news is the sorting function call appears optimised to the same level as standard sorts, up until the point your code takes over! Taking the simplest of standard sorts:

```
MyQueue         QUEUE
SomeLong             LONG
            END

...

  SORT(MyQueue, +MyQueue.SomeLong)
```

And replicating it as a functional sort:

```
CompareMyQueue  GROUP(MyQueue),TYPE
                END

...

  SORT(MyQueue, MySortFunction)

...

MySortFunction  procedure(*CompareMyQueue a,*CompareMyQueue b) ! ,signed

    code

    if a.SomeLong = b.SomeLong
      return 0
    elsif a.SomeLong < b.SomeLong
      return –1
    else
      return 1
    end
```

The results are excellent. Perhaps surprisingly, some quick tests suggest that the functional sort may even be slightly faster. But there is no `WHAT()` trickery going on above, those are straight comparisons of `LONG`s. Problems start when writing generic functions that involve a lot of code and/or slow operations. The function will be called many, many times during a `SORT`/`ADD` etc and time spent inside it becomes a significant factor.

While optimisation of that code will obviously help, the sheer number of calls made mean

that slow sorts may be inevitable. The function used by the class (described in Part 2) is an example – despite some effort at optimisation it is on the slow side. It seems there is sometimes a trade-off to be made between the speed of standard (or dedicated functional) sorts and the flexibility a generic sort function offers. Incidentally, it's a very uneducated guess but looking at the number of times any function (generic or dedicated) is called suggests a bubble sort may be in action (you may wish to read Alison Neal's article on the subject).

Problem #2 - Clarion may refuse to call your sort function

Surely not!

Sorry, I'm afraid it's true. This brings us back to sort-specific instructions. Staying with the sort shown above, standard sorting also allows reversing the order:

```
SORT(MyQueue, -MyQueue.SomeLong)
```

How to get the same functionality from that single sort function? Seems easy enough - just have a flag external to the function indicating the direction of sort (+1 or –1) :

```
MySortDirection LONG(+1) ! initialize to "forward"

...

MySortFunction  procedure(*CompareMyQueue a,*CompareMyQueue b) ! ,signed

    code
    if a.SomeLong = b.SomeLong
      return 0
    elsif a.SomeLong < b.SomeLong
      return –MySortDirection
    else
      return MySortDirection
    end
```

and when the direction of sort (stored in `MySortDirection`) is changed, simply:

```
SORT(MyQueue, MySortFunction)
```

But in general that will fail. Clarion remembers the queue was last sorted by a call to `MySortFunction` and if the queue contents are unchanged it does you the favour of avoiding what it believes to be an unnecessary sort.

If a user action is to influence the sort order of a queue then this "intelligence" on Clarion's part gets in the way. And there is no documented way to force the sort to happen. For me, this came to light as a result of writing exploratory code for the class – my user action being mouse clicks on column headers. After some head-scratching I asked those nice people on the newsgroups for help. There were two solutions.

1. Fool Clarion into thinking the queue has changed.

This makes sense but it took a while for the first nice person (Jeff Slarve) to arrive at a combination of `GET/ADD/DELETE` that seem to always work:

```
GET(MyQueue, 1)
ADD(MyQueue, 1)
DELETE(MyQueue)
```

2. Use a function from the RTL

Turns out there is a Run Time Library function `Cla$FREEQueueFKey` to "reset the queue keys" – available in C6 and C7. Thanks go to Robert Peros and Jonathan Kay. This was the approach I used within the class and template. The template adds the RTL procedure to the Global Map:

```
module('clarionrtl')
  SDKLCSSortFreeRTL(*queue, *SDKLCSCompareFunc),NAME('Cla$FREEQueueFKey'
end
```

And within the class, prior to calling a functional sort, the queue sort is reset:

```
SDKLCSSortFreeRTL(self.listfrom, SDKLCSCompareFunc)
sort(self.listfrom, SDKLCSCompareFunc)
```

The RTL function is not available in C5/C55.

During early research and test coding, this appeared to be the last piece of the jigsaw. I knew a functional sort would do the job and with the RTL magic it could all be pulled together. OK, there was the daunting prospect of writing a template to make it easier to use the class, but it all seemed possible.

In Part 2 I'll cover the class and the template.

Download the source

Simon is an independent developer based in the UK. Since the late 80s he has spent most of his time aping real Clarion programmers.

Article comments

by Bob Roos on August 8 2010 (comment link)

Just in time. I just had the need for a case insensitive sort and didn't want to duplicate the fields. Thanks. (Now I just have to wait for Part II)

by Rick Horn on August 10 2010 (comment link)

Good Article. Who wrote it?

by Dave Harms on August 11 2010 (comment link)

Whoops. Thanks for pointing that out, Rick. Simon's bio and, um, pic attached now.

Dave

---

by Simon Kemp on August 11 2010 (comment link)

um?? That's my best side.

---

by Dave Harms on August 11 2010 (comment link)

Download link fixed - my apologies.

Dave

BACK TO TOP

Custom Queue Sorting, Part 2

By Simon Kemp

Posted August 12 2010

In Part 1 I introduced functional queue sorting and showed a few sort functions. Now it's time to look at the class and the template.

The class and template

My objective was a quick and easy way to replicate Clarion's "additional browse sort headers" when using simple queue-driven list controls (Clarion's template is only available as an extension to table-driven browse boxes).

Perhaps unusually, I find myself placing queues on windows pretty often. Then it can be necessary or useful to offer the familiar click on header functionality. This always came down to hand-coding - not too tedious but ultimately error-prone.

The handcoding approach goes like this:

- Alert relevant keycodes on the list box.
- Detect the keycodes, check the header is being clicked.
- Note the column, call a routine / procedure tied to the queue and list control.
- Decide whether the sort is new or being "reversed", `SORT()` the queue, display an indication of current sort in the headers, maybe remember the sort column/direction for next time window opens.

It goes on but you will get the idea.

All of this is involved enough that I had never found the motivation to attempt proper multi-column sorts (in the sense of an indication in the headers that "column 2 is sorted within column 1 within column 3"). But if the whole deal could be templatised it would justify going that extra mile in the (re-usable) code.

Four files are associated with this article; the installation is manual (no installer).

The class source code files are sdklcs.inc and sdklcs.clw. Either keep these local to the demo app or drop them into the relevant add-on libsrc

The In-Memory Database Driver

What about the IMDD? If I switch

folder (C6: clarion\3rdParty\libsrc, C7: clarion\accessory\libsrc\win\)

The template is in sdklcs.tpl - register it manually in the IDE.

The demo app (sdklcsdemo.app) is made with C6 (and works with C7), but no restriction to ABC is implied for the class or the template.

The template

I'll approach this "top-down", starting with the template. There is a global extension (Figure 1) that enables the class. All it's doing is including sdklcs.inc/ sdklcs.clw, adding some other stuff (mentioned previously) to the global map, and getting things ready for when the procedure template is associated with a list (or combo) control within a window procedure.

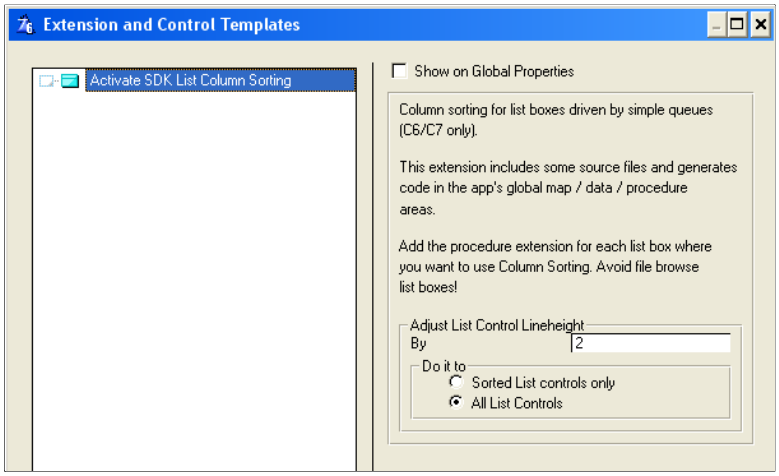


Figure 1. Global Template

There is just one option here – a facility to tweak the list control's `PROP:LineHeight` across all windows in the application. Figure 1 is from the demo app and I have chosen to increase lineheight by 2 for all list controls. That "Sorted List controls" option means restrict the line height adjustment to controls where the procedure template is also in use. I suspect neither option will prove useful, however they were easy to add so are waiting just in case.

Getting down to the nitty-gritty, Figure 2 shows the procedure extension for a list control in the demo app (it's the one in the bottom half of the first tab, ?List2, driven by local queue Queue2).

from using Queues to in-memory tables none of this is a problem because I can use all the standard Clarion templates. Isn't that one of the great things about the IMDD? Well yes, if you have never used queues in this way and feel you never will then the IMDD should work just fine.

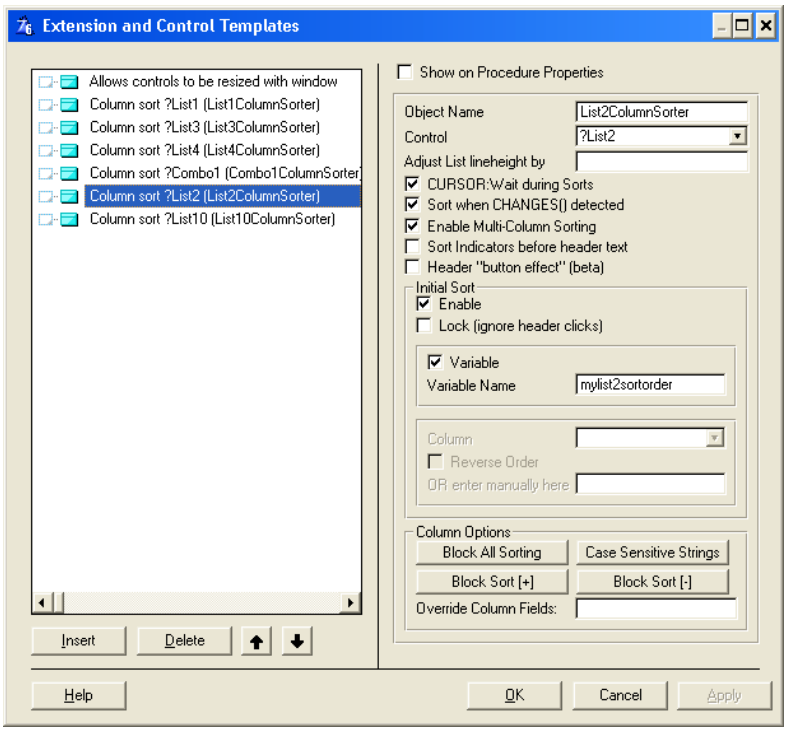


Figure 2. Procedure Template Main Options

Working through the prompts:

Object Name: the template will suggest something unfriendly for the class instance such as `ListColumnSorter27`. It's overriden here with something more obviously related to the control (`List2ColumnSorter`).

Control: initially blank, `?List2` was chosen from the drop-down of candidate controls.

Adjust List lineheight by: can be used to tweak prop:lineheight on a control by control basis, overriding any setting in the global extension.

CURSOR:Wait during Sorts: on by default, meaning it shows the hourglass cursor when a queue sort is triggered.

Sort when CHANGES() detected: on by default and instructs the class to keep track of the queue content using Clarion's `CHANGES()` function. If queue content is changed in code (or perhaps using EIP) the class will trigger a sort. This can only happen the next time a trip round the `Accept` loop causes the class `TakeEvent` method to fire. In some circumstances you will want to force it and one way is calling the `TakeEvent` method in your code. Alternatively, just call the `Sort` method (that's what `TakeEvent` does when it detects a change in queue content).

Enable Multi-Column Sorting: on by default. Mimics Clarion's additional sort header

interface - i.e: `LeftClick` for single column sorts, `CtrlLeftClick` for additional sort orders, `ShiftLeftClick` to forget any sort(s).

Sort Indicators before header text: off by default. Select it to place the indicators ([+], [-], [+2] etc) *before* any existing column header text.

Header Button Effect: off by default. Turn this on to replicate Clarion's own “button press” effect when headers are clicked.

Enable Initial Sort: off by default. Turn this on if the control's queue should be sorted as the window opens.

Lock Initial Sort: off by default. Turn this on if the initial sort needs to remain "fixed". Indicators will be placed in the column headers but all user interaction to modify sort order is disabled.

Variable Initial Sort: specifies that a (string) variable supplies any initial sort. Here mylist2sortorder is a local string of 10 characters, initialised to '+5-6'. Using a variable in this way also gives scope to make a user's choice of sort order "sticky" because the class updates the variable as changes in sort order are made. So by declaring the variable as static any previous sort selection will remain in place next time the window opens.

The other option (enabled by turning off the 'Variable' checkbox) is to simply fix the initial sort to one or more columns. The template allows you to choose a single column from the dropdown list or (if multiple columns are involved) you can enter the sort order manually: for example enter '+2-3' for "column 3 sorted in reverse order within column 2".

Finally comes the group of column options, the last of these is:

Override Column Fields. This is provided to cater for occasions when the displayed column content is not suitable for sorting. It tells the class to use a different column when that header is clicked. See the fifth tab of the demo app for an example. Typically the “override column” would be hidden by setting it's width to zero. Building a nice template dialog for this setting was beyond me! You are stuck with entering a Clarion expression such as '001002003004'. Column numbers and their overrides are specified as pairs of n03 strings. This example ('001003002004') means column 1 is overridden by column3 and column 2 is overridden by column 4.

There are also four column option buttons and associated dialogs.

- Block All Sorting: mark any columns that should never be included in sorts.
- Block Sort [+]: mark any columns where only reverse sort is permitted.
- Block Sort [-]: mark any columns where only forward sort is permitted (if the same

column is tagged for 2 and 3 it has the same effect as 'Block All Sorting').

- Case Sensitive Strings (Figure 3). I've deliberately left this until last.

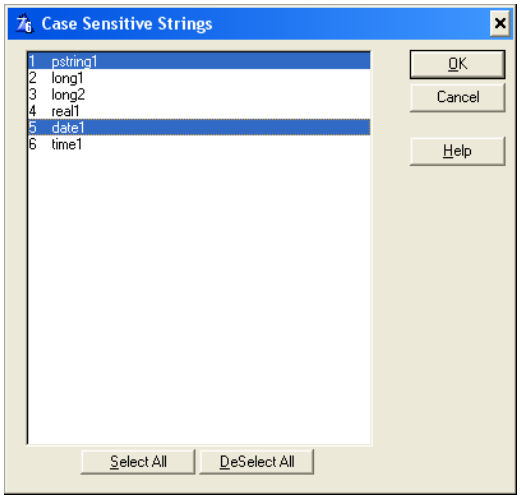


Figure 3. Procedure Template Case Sensitive String dialog

By default, the class sorts strings with case-sensitivity OFF. Here is the declaration of Queue2 along with a picture of the list control (?List2):

```
Queue2                QUEUE,PRE(q2)
pstring1                PSTRING(40)
somegroup               GROUP,PRE()
long1                     LONG
long2                     LONG
real1                     REAL
                        END
anothergroup            GROUP,PRE()
date1                     DATE
time1                     TIME
                        END
                      END
```

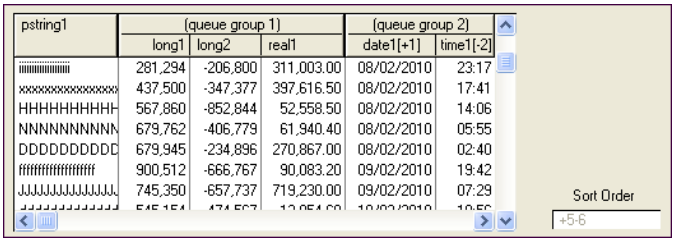


Figure 4. List control in the demo application

In Figure 4 the template dialog permits column 5 (`Queue2.date1`, a `DATE`) to be marked as case-sensitive. This has no meaning and in fact is ignored by the class. The reason the class

knows it is irrelevant leads nicely into the detail of how a generic functional sort deals with a structure such as `Queue2`.

In the third and final installment I'll cover the use of the `WHAT()` function, the class design, and the final sort function.

Download the source

Simon is an independent developer based in the UK. Since the late 80s he has spent most of his time aping real Clarion programmers.

Article comments

BACK TO TOP

Custom Queue Sorting, Part 3

By Simon Kemp

Posted August 13 2010

In Part 1 of this series I introduced the concept of queue sorting using custom functions, and in Part 2 I described the class and the template. Now it's time to look at the role of the `WHAT()` function, explore the class design, and review the final sort function.

Using WHAT

First, there appear to be those groups within the queue to contend with. Any generic access to the queue requires the use of `WHAT()`. Although `Queue2.date1` is column 5 in the list control, `WHAT(Queue2, 5)` will reference `Queue2.real1`. This is because the start of a group counts as a field. For the functional sort to access `Queue2.date1` it needs to consider the two groups and use `WHAT(Queue2, 7)`. This looks problematic but is actually a non-issue. Relying on the list column to access the queue structure would be the wrong approach because fields from the queue can of course be positioned in any column. Fortunately Clarion provides `PROPLIST:FieldNo`. While the class is very much driven by list control columns, the link into queue structures is made using `PROPLIST:FieldNo` (an array):

For column 5, the header text can be retrieved using `?List2{PROPLIST:Header,5}` (initially returning the string `'date1'`) and the critical link into the queue structure results from `?List2{PROPLIST:FieldNo,5}` returning 7. The value 7 can then be used with `WHAT` (or indeed `WHO` if the label within `Queue2` is required). This provides access to the data "behind the column" and it all starts to come together. In fact, if sorting on a single column was the extent of our ambition the job appears done.

To sort forwards on column 5, just set two (`LONG`) parameters accessible from the sort function:

```
MySortDirection = +1
MySortField = 7
```

Surely the sort function can then be this simple:

```
MySortFunction  procedure(*group a,*group b) ! ,signed

    code
```

```
    if WHAT(a, MySortField) = WHAT(b, MySortField)
      return 0
    elsif WHAT(a, MySortField) < WHAT(b, MySortField)
      return –MySortDirection
    else
      return MySortDirection
    end
```

Not quite. This does work - but only for numeric data types such as `LONG`, `DATE`, `TIME`, `SHORT`, `REAL` etc. Strings require assigning to an `ANY`. It does no harm to do the same with numeric data types so Version II looks like this:

```
MySortFunction  procedure(*group a,*group b) ! ,signed

elementa  ANY
elementb  ANY

    code

    elementa &= WHAT(a, MySortField)
    elementb &= WHAT(b, MySortField)
    if elementa = elementb
      return 0
    elsif elementa < elementb
      return –MySortDirection
    else
      return MySortDirection
    end
```

If there were no requirement to remove case-sensitivity from string sorts then Version II could form the basis of the sort function used by the class. That requirement turns out be more challenging than it might appear. Ignore the fact that the final intention is to make case-sensitivity optional – surely it should be easy to ensure all strings sorts are case-insensitive (i.e. `UPPER()` is applied across the board). Taking Version II and modifying the two comparisons:

```
if UPPER(elementa) = UPPER(elementb)
  return 0
elsif UPPER(elementa) < UPPER(elementb)
  return –MySortDirection
else
  return MySortDirection
end
```

The above code works for strings but breaks with numeric data types. `UPPER` returns a string so `UPPER(123)` is less than `UPPER(20)`. OK, next thought would be using `UPPER` on those `ANY`s.

```
elementa &= WHAT(a, MySortField)
elementb &= WHAT(b, MySortField)
elementa = UPPER(elementa)
elementb = UPPER(elementb)
```

That should have no effect on numeric data types and give the required result with strings. But it's a very bad idea indeed. By doing this, string elements in the queue structure are *typically* written back into the queue in a changed state ('Simon' permanently becomes 'SIMON'). But there is a mysterious wrinkle here because on occasion it looks to be a safe approach. I must confess to not pursuing this in any detail. Hopefully it was a mistake on my part or related to a particular queue structure / sort order. For efficiency it seems unlikely the groups would ever be 'copies' of the queue entries yet sometimes it seemed safe to modify them. But moving on - intuitively and in practice it seems wrong to modify the groups so another approach to dealing with strings is required.

The nub of this problem is knowing when it is safe to use the string comparison:

```
if  UPPER(elementa) = UPPER(elementb)
```

Thankfully Clarion's `ISSTRING()` provides the necessary information. Version III requires just one more piece of external information to safely achieve case-insensitive sorting:

```
MySortDirection = +1
MySortField = 7
MySortIsString = ISSTRING(WHAT(SomeQueue, MySortField)) ! FALSE

MySortFunction  procedure(*group a,*group b) ! ,signed

elementa  ANY
elementb  ANY

    code

    elementa &= WHAT(a, MySortField)
    elementb &= WHAT(b, MySortField)
    if MySortIsString
      if UPPER(elementa) = UPPER(elementb)
        return 0
      elsif UPPER(elementa) < UPPER(elementb)
        return –MySortDirection
      else
```

```
      return MySortDirection
    end
  end
  if elementa = elementb
    return 0
  elsif elementa < elementb
    return -MySortDirection
  else
    return MySortDirection
  end
```

Note the use of `ISSTRING(WHAT(SomeQueue, MySortField))` above. The help for `ISSTRING` states:

> ISSTRING(field)
>
> ISSTRING Returns true if the field is a STRING, CSTRING, or PSTRING data type.
>
> field The label of a field.

This usage of `WHAT` does not quite make sense to me – it is certainly not returning "the label of a field". As I have shown, `WHAT` can be used for straight comparisons of structure elements:

```
if WHAT(a, MySortField) = WHAT(b, MySortField)
```

or assigning an element to an `ANY`:

```
elementa &= WHAT(a, MySortField)
```

But using it with `ISSTRING` in this way turns out to work perfectly, identifying all `STRING`, `CSTRING` or `PSTRING` elements in a queue's structure. Thank you Clarion!

Class design

Central to the class is a queue detailing the columns of the list or option control:

```
sdkLCScolq      queue,type
column            long
columnwho         string(100)
headertext        string(255)
sortoptions       long
fieldno           long
isastring         long
                end
```

Its elements should now be almost self-explanatory:

column	the column number
columnwho	the label of the queue field, retrieved using WHO() but not used at present
headertext	the initial header text for the column (PROPLIST:Header)
sortoptions	a bitmap detailing case-sensitivity and those 'block sorting' options
fieldno	link into the control's queue (PROPLIST:FieldNo)
isastring	is fieldno a string?

This queue (the 'column queue') is built during initialisation of the object and it's content does not change.

There is a second queue (the 'key queue') to describe the current sort:

```
sdkLCSkeyq      queue,type
level             long
column            long
direction         long
fieldno           long
sortoptions       long
isastring         long
                end
```

This queue is used to drive the functional sort. It is updated for two reasons – when an initial sort order is passed during class initialisation or as a result of user interaction. If no sort is active the queue will be empty. If multi-column sorting is disabled it will never contain more than one entry. The first two elements (level and column) are not used by the sort function but required by class methods to track, modify and report on the current sort order:

level	1, 2 etc
column	pointer back to the column queue
direction	+1 for forward, -1 for reverse (Version III's MySortDirection)
fieldno	copied from the column queue (Version III's MySortField)
sortoptions	copied from the column queue
isastring	copied from the column queue (Version III's MySortIsString)

At this point it is possible to look at the sort function in abstraction. By design, a key queue will contain all it needs. So let's get that out of the way.

The final sort function

I failed to come up with a way of embedding a functional sort in the class itself. Perhaps that can be done but it is well outside my Clarion coding envelope! However, because this is a generic function only one is actually required. So the code is included in sdklcs.clw (it becomes a global procedure) and the template adds this prototype to the global map:

```
SDKLCSCompareFunc(*group, *group),signed
```

All well and good. But having done that, multiple instances of the class potentially running on different threads must ensure the function is looking at their key queue before asking Clarion to run a sort. I suspect I broke all best practice regarding encapsulation at this point but needed to get things working. The template also adds this to global data:

```
SDKLCSkeyqref  &sdkLCSkeyq,thread
```

The relevant three lines of code in the class `Sort` method, including that magical RTL call, become:

```
SDKLCSkeyqref &= self.keys
SDKLCSSortFreeRTL(self.listfrom,  SDKLCSCompareFunc)
sort(self.listfrom, SDKLCSCompareFunc)
```

Here is the sort function code. It refers to the global key queue reference for guidance on how to compare the groups passed by Clarion:

```
SDKLCSCompareFunc procedure(*group cfTarget, *group cfCompare) ! ,signed

myPtr       long
myanyt      any
myanyc      any

    code

    loop myPtr = 1 to records(SDKLCSkeyqref)
      get(SDKLCSkeyqref, myPtr)
      myanyt &= what(cfTarget, SDKLCSkeyqref.fieldno)
      myanyc &= what(cfCompare, SDKLCSkeyqref.fieldno)
      if SDKLCSkeyqref.isastring and ~band(SDKLCSkeyqref.sortoptions,2^1)
        ! case-less string sort required
        if upper(myanyt) < upper(myanyc)
          return -SDKLCSkeyqref.direction
        elsif upper(myanyt) > upper(myanyc)
          return SDKLCSkeyqref.direction
        end
      elsif myanyt < myanyc
        return -SDKLCSkeyqref.direction
```

```
      elsif myanyt > myanyc
        return SDKLCSkeyqref.direction
      end
    end
    return 0 ! all keys are equal
```

It's a kissing cousin of Version III. A loop through the referenced key queue has been added, plus one extra piece of information regarding string case-sensitivity and it does the job well.

Let's wave bye-bye to functional sorts for now and wind-up with a quick look at the detail of class usage.

Manual use of the Class

The procedure template is designed to be all you need. Just fill in the prompts and it does all the work. But it does rely on two things:

a. The control needs a known queue structure behind it (`PROP:From`).
b. The list or combo control needs format information (`PROP:Format`) available. This is used to gather information about the control using the `PROPLIST:Header` and `PROPLIST:FieldNo` arrays.

But there are times when one or both will not be part of the window design because a developer plans to specify them in code at run-time.

It could be because they need to display a referenced queue. In this case `PROP:Format` would usually be defined but `PROP:From` set to '' at design time. In code, the queue would then be assigned to the control using `?MyListBox{PROP:From} = somequeuereference`.

A more extreme case would be a list control intended for use as a generic "queue viewer". `PROP:Format` would be constructed at run-time by interrogating the queue structure and again `PROP:From` used to assign a queue to the control.

In either case the template does not have a chance. It will blindly go ahead and try, but the generated code will cause compiler errors – in particular when it attempts to pass `PROP:From ('')` as a queue.

To use the class, you must wait until the `Queue` to be used is known and the list control's `PROP:Format` is available. Declare an instance of the class:

```
MyColumnSorter  sdkLCS
```

Ensure this code is somewhere at the top of the accept loop:

```
if MyColumnSorter.TakeEvent()
  cycle
end
```

Initialise and `Start` the class. This is a slightly messy process, potentially involving both methods and properties. A fairly simple example would be:

```
! Initialise (or reset) the class, more settings may follow
MyColumnSorter.Init(?MyDynaList, SomeQueueLabel)
! Block all sorting on column 1
MyColumnSorter.SetColumnOptions('10000000', '', '', '')
! Override default indicator positioning
MyColumnSorter.prependindicators = true
! pass a variable (initially blank) to receive details of sort order
MyColumnSorter.SetInitialSort(MyStickySort)
! All settings done, “start” the class (carrying out any initial sort)
MyColumnSorter.Start
```

This is demonstrated in some detail on the third tab of the demo application, where the nine queues used elsewhere in the application are dynamically assigned to a list control. Search the code for the `QueueBrowser` procedure to see how this is done.

In fact I quickly realised that some of those queues needed to be identical copies to avoid two instances of the class fighting over the same queue! Taking a quick look at the local data definitions and the `BuildQueues` procedure will reveal what is going on.

Conclusion

Functional queue sorting is a powerful feature lurking rather quietly in the Clarion toolkit. Not something you will need every day (or month) but when a difficult in-memory sorting problem arises it will often be the answer.

Here it has been used as the basis of a useful class to support column sorting in queue-driven list controls.

Download the source

Download the Sept 22 2010 update

Simon is an independent developer based in the UK. Since the late 80s he has spent most of his time aping real Clarion programmers.

Article comments

by Bob Roos on August 14 2010 (comment link)

THANK YOU Simon.

That is super useful and comes just in time for me. I struggled with it a bit because it was too easy. In my case just add the global template and then add the extension in the

procedures with the list boxes I wanted to sort. That was all! I kept looking for more to do.

---

by Simon Kemp on August 15 2010 (comment link)

Thanks for that Bob - glad you like it.

---

by mark bessemans on September 17 2010 (comment link)

Thank you! Great example, very well explained.

---

by Dave Harms on September 22 2010 (comment link)

I've posted an updated source zip supplied by Simon (see the link above) - this version supports C5.5, among other improvements.

Dave

---

by Simon Kemp on September 22 2010 (comment link)

Background to that c55 update is someone more knowledgeable than me pointing out that rtl support was provided prior to c6. Digging, appeared this was done to support the ABC additional sort headers feature because looking at the top of [c55] \libsrc\abdrops.clw you will find a subtle rtl reference: Cla$ADDqueuefkey. Just above it, two interesting prototypes:

```
CaseSensitiveCompare (*GROUP,*GROUP),SIGNED
CaseInsensitiveCompare (*GROUP,*GROUP),SIGNED
```

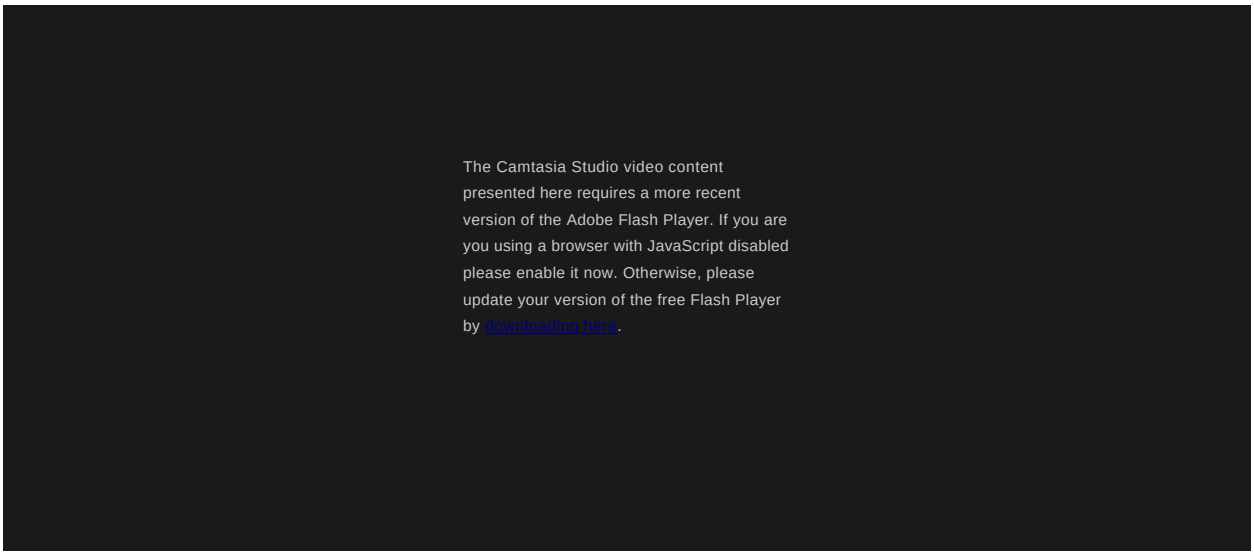
Bingo. And libmaker on c55runx.dll then revealed the 4 expected queue access functions (add/get/put/sort) in line with c6:-

Cla$ADDqueuefkey Cla$GETqueuefkey Cla$PUTqueuefkey Cla$SORTqueuefkey

There was however no magical "reset functional sort" call - seems this had to wait until c6. So to reset the queue sort in c55 (see part 1 of the article) it was necessary to use the "nice person" brute force solution.

Suspect this is all a bit academic, but wish I'd known it was available a few years back when fighting queues in c55 :-)

BACK TO TOP


The Camtasia Studio video content
presented here requires a more recent
version of the Adobe Flash Player. If you are
you using a browser with JavaScript disabled
please enable it now. Otherwise, please
update your version of the free Flash Player
by downloading here.

Microsoft LightSwitch: Impressive, but not a tool for building big apps

By Dave Harms

Posted August 26 2010

Earlier this month I reported with some anticipation on Microsoft's upcoming beta release of LightSwitch, the company's application development environment for Silverlight line of business applications (both web and desktop).

I've had some time to play with LightSwitch, and it is an impressive product. At this point, however, I don't think it's something you can use to build large .NET business applications, although it could certainly become that kind of tool in the future.

The critical issue in LightSwitch is this: it's an application generator, not an application *code* generator. That is, it doesn't generate actual source code (or not much); rather, it creates a model of the application, and then it interprets that model at runtime.

In contrast, Clarion (yes, I'm talking Win32 Clarion) generates Clarion source code which gets compiled into an application.

LightSwitch's runtime interpretation approach is closer to the old CPD pseudocode, although it's even more abstract than that. And there are advantages to using an interpreted runtime model - for one thing, LightSwitch lets you modify the running application, sort of like an application-wide screen preview. Hiding the implementation is also an advantage for Microsoft - there's no way developers can modify generated source and have their changes accidentally wiped out on the next generation cycle, and Microsoft is also completely free to change the internal architecture of the application.

Black box problems

But there are some important downsides to LightSwitch's model-based approach as well. It turns the application into a system. Not only do you not know what's going on inside the black box, you can't reuse that code. And in the .NET world, much more so than the Clarion Win32 world, code reuse is essential.

It's a shame you don't get code, because a LightSwitch app does some pretty nifty things. It's a multi-tier application with a Silverlight client front end, a middle tier that handles the

business logic, a back end database (typically SQL), and all the plumbing to make the different tiers work together. You can run the Silverlight app in a browser, as a rich internet application, or outside the browser as a desktop app.

Unfortunately, you don't get to see how all of that code works, because there isn't any source. You can't learn from it, and you can't reuse it in another application; each LightSwitch application is an island unto itself. You can plug your own custom code into a LightSwitch app, but you can't reuse a LightSwitch screen, or the data layer, or business and validation rules in another LightSwitch app or a non-LightSwitch app.

I don't see how you would do the equivalent of a multi-DLL Clarion app in LightSwitch (with one app as the data DLL), at least not without a whole lot of duplicated code. And once you start duplicating apps you begin to lose much of the advantage LightSwitch offers.

In a nutshell, LightSwitch's level of abstraction is extremely high. That makes it a good end-user tool, and suitable for lightweight, small app development. In its current state it's almost certainly going to cause a world of pain if used for Clarion-style full-on business app development.

So what needs to change? In a nutshell, LightSwitch needs to be able to generate code. (And logically, it should do it using T4, Microsoft's own template language.) Generating code would yield the following benefits:

- Reusability - there would be actual assemblies that could be reused elsewhere
- Visibility - developers could learn from, and better interact with, the overall application architecture
- Upgradability - if for any reason a LightSwitch app proved insufficient it could be taken over as pure source and enhanced.
- Security - in the unlikely event that MS abandons LightSwitch, there's always the code.
- Customization - if the template system is exposed then anyone can create or customize the templates. This would be a great way to leverage community knowledge about how business apps should be designed.

Will LightSwitch generate code one day?

In version 1 LightSwitch will definitely *not* generate source code. But if there's enough interest, that capability might be added down the road. I haven't heard any promises, but the LightSwitch team is actively soliciting feedback and suggestions. In fact, the general attitude from the Microsoft developer tool teams these days seems to be "What would you like to be able to do? How can we help you?"

That's refreshing.

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

Article comments

by Jerry Walters on August 27 2010 (comment link)

Are there any size benefits to this interpreted vs actual code model? Have we gone back to the days (daze) of use once, re-create if ever needed again? Sounds almost as bad as .Net, you can do anything, here's thousands of pieces, some assembly required.

by Dave Harms on August 27 2010 (comment link)

Jerry,

The architecture embodied in a LightSwitch app is essentially three tier so there's more going on than in a simple client/server app. How much of that is standard .NET code and how much is specific to LS, I can't say.

Is it a step back? In terms of reuse, yes. In terms of architecture, it's definitely a big step forward.

Dave

by luuk sluyter on August 30 2010 (comment link)

I had not heard of LiteSwitch, before you wrote about it, so I also tried it out. I created a very simple application with 3 tables and one createNew screen, and already ended up with a project consisting of 760 files in 145 folders with almost 80 Mb of code... Hidding away complexity from the user, creates much more complexity at the applicationsite, and though sourcecode is available it is very hard to manualy change anything there. One of the charms of Clarion is its efficiency and the way you can leave out what you don't need. I hope the Clarion.net appgen will be a released soon!

Best regards Luuk Sluijter

by Dave Harms on August 30 2010 (comment link)

Luuk,

Almost all of the that 80MB of files can be found under ClientGenerated\Bin\Debug and consists of different versions of the client app for different locales. Not the most efficient way of handling localization, but there you have it.

But the complexity is definitely there - have a look at this Channel 9 video on the architecture, especially the UI stuff toward the end.

A purely desktop WPF (or WinForms, if anyone's still starting new dev using that technology) app wouldn't need that same level of complexity. But you also wouldn't be able to deploy as a web app if you wanted.

It's an interesting choice with a lot of benefits, but there are also some potential costs in

performance (slower), features (fewer) and complexity.

BACK TO TOP