Clarion Magazine

This edition includes all articles, news items and blog posts from April 1 2011 to April 30 2011.

Clarion News

Read 12 Clarion news items.

The ClarionMag Blog

Read 3 blog entries.

Articles

Tip of the Week: Forcing Code Completion

April 7 2011

If, like Dave Harms, you've become dependent on code completion, you've also probably noticed times when it doesn't conveniently pop up. Here's how you can (usually) force code completion to do your bidding.

Tip of the Week: Window Previewer Tricks

April 15 2011

Yes, there's a window previewer in C7, just like in C6. Well, not just like in C6. Here's how to use the C7 previewer's runtime advantage.

April articles still to come!

April 19 2011

Here's a look at some of the April articles appearing shortly.

Tip of the Week: Customizing the Window Previewer's Data

April 22 2011

The C7 window previewer supplies dummy data for controls, making it easier to see what the window really will look like when it's in a running app. Here's how you can customize that data.

Getting Rid Of "[__Dont_Touch_Me__]" INI Files

April 24 2011

When you create a new application, Clarion assumes you want an INI file. But even if you don't tell your app to use it, it will be there with that obnoxious "[__Dont_Touch_Me__]" entry. Steve Parker shows how to get rid of the INI file once and for all.

Filtering Hand Coded Lists Using The TREE Attribute

April 25 2011

There are lots of ways to filter hand coded listboxes, and they usually involve creating a separate display queue. But as S. Jayashanker explains, you can use a single queue and filter records for display using the TREE attribute.

ClarionMag's Spring Sale Ends Wednesday, April 27

April 25 2011

Subscription sales have become a rarity at Clarion Magazine. But it's been a long and grueling winter up here on the Canadian Prairies, and with spring finally taking hold it seems like a good time to do some celebrating! Save up to $62!

In Memory Of Nigel Hicks

April 28 2011

Nigel Hicks, one of the London Clarion developers, has passed away.

How To Have Your [Browse] Cake And Eat It Too (a refreshing story)

April 29 2011

John Morter suspected his browses were sometimes being loaded twice; his curiosity about what was really happening led him on a Dr. Parkeresque journey of discovery.

SV Announces 2011 Clarion DevCon

April 29 2011

SoftVelocity's Bob Zaunere has announced there will be a 2011 Clarion DevCon. This event is being co-organized by SoftVelocity and ClarionLive, and will be held at the Inverness Hotel and Conference Center in November in Denver, Colorado.

A String Class

April 29 2011

It's a bit over eight years since Steve Parker wrote his first class. Here's how he was recently motivated, no, forced to write his second.

Clarion 8: First Look

April 30 2011

Clarion Magazine's David Harms takes a tour through the first Clarion 8 beta and finds much to like.

Clarion News

Clarion.NET Update

SoftVelocity has posted an update on Clarion.NET, indicating steady progress on AppGen and reporting the major delay has been creating T4 equivalents to the unique Clarion template language statements such as #Procedure, #Code, #Group, #Embed etc. The blog post also indicates future support for Android and iOS.

Posted April 13 2011 (permanent link)

XII DevCon Brasil Video

A short video about the upcoming XII DevCon Brazil is available on YouTube. The beginning of the audio is in Portuguese and English is at the end.

Posted April 15 2011 (permanent link)

ChartPro Wrapper Template 1.01

Version 1.01 of the Noyantis ChartPro Wrapper Template is now available. This release contains the following updates: Class modified to add Thin@ Support; The control can now be added to the main AppFrame multiple times, and in multiple positions; Two new methods added to help with third party compatibility. The update is free to all users who have an active Maintenance Plan in place, or costs $47.50 for all users who have a lapsed Maintenance Plan. It can be downloaded via the Products page within the members area of the web site.

Posted April 15 2011 (permanent link)

SuiteControls Wrapper Template 1.04

Version 1.04 of the Noyantis SuiteControls wrapper template is now available. This release contains the following updates: Codejock 13.4.1 -> 15.0.2 compatibility added; SuiteControls class introduced to replace individual item classes of PopupControl Class, TabControlClass Class, TaskDialogClass, WebBrowserClass; Class modified to add Thin@ Support; Two new methods added to help with other third party compatability; The SheetEnhancer now allows for the contents of the current tab to be automatically moved to keep a relative position to the tab bar when in "Visio" mode; New SheetEnhancer Options added; Several bug fixes. The update is free to all users who have an active Maintenance Plan in place, or costs $47.50 for all users who have a lapsed Maintenance Plan. It can be downloaded via the Products page within the members area of the web site.

Posted April 15 2011 (permanent link)

CodeJock ShortcutBar Wrapper Template 2.00

Version 2.00 of the Noyantis Codejock ShortcutBar Wrapper Template is now available. Changes include: Codejock 13.4.1 -> 15.0.2 compatibility added; Multi Threading Enhanced; OCX Registration Enhanced; Support for Visual Theme Resource files added. (Office 2007, 2010 + Win7); The control can now be added to the main AppFrame multiple times, and in multiple positions; Template Optimized; New Class Implemented to implement the template settings; Icons now automatically added to Project List; Trappable OCX Events can now be added and removed at runtime; Trappable Keystrokes can now be added and removed at runtime; Tooltips added to individual Bars; Bar Resize Strategy added; New setting added : "Allow Resize By Gripper"; New setting added : "Enable Animation"; New setting added : "Show Active Bar on Top"; New setting added : "Show Expand Button"; New setting added : "Select Default Option on Bar". In this build as much generated code as possible has been moved into the class. This makes it easier to implement the control (eg, in a source procedure) and also allows for greater flexibility with other third party addons. All of the previous embed points are still valid and have been automatically remapped into the new derived methods. Now, there are no local procedures generated for you by the template - only derived methods. This does mean however that if you are manually calling any of the previous local procedures, these calls should now be replaced with a call to the new related class method. The update is free to all users who have an active Maintenance Plan in place, or costs $47.50 for all users who have a lapsed Maintenance Plan. It can be downloaded via the Products page within the members area of the web site.

Posted April 15 2011 (permanent link)

ShortcutBar Wrapper Template Demo

A new ShortcutBar Wrapper Template demo app is available from Noyantis.

Posted April 15 2011 (permanent link)

Clarion 7.3.8222 Released

Clarion 7.3 build 8222 includes a number of bug fixes and changes. For those working in shared directories, the procedure wide embed editor now uses a unique temporary filename for each process.

Posted April 20 2011 (permanent link)

IP Driver for Clarion 7.3.8222

A build of the IP driver is now available for Clarion 7.3.8222.

Posted April 20 2011 (permanent link)

EasyExcel 4.06

EasyExcel 4.06 is now available.

Posted April 25 2011 (permanent link)

ShortcutBar Wrapper Template 2.01

Version 2.01 of the Noyantis ShortcutBar wrapper template is now available. This release contains the following updates: Class modified to add Thin@ Support; The control can now be added to the main AppFrame multiple times, and in multiple positions; Two new methods added to help with other 3rd Party compatability. The update is free to all users who have an Active Maintenance Plan in place, or costs $47.50 for all users who have a Lapsed Maintenance Plan. It can be downloaded via the Products page within the members area of the web site.

Posted April 25 2011 (permanent link)

PrintWindow 2.02

Changes in PrintWindow 2.02 include: Several options to fill checkboxes and radio buttons on print.

Global option to print Regions borders. If you don't own PrintWindow, a new copy will cost U$S 99 to you. No subscription or maintenance fees.

Posted April 25 2011 (permanent link)

CommandBars Wrapper Template 2.08

Version 2.08 of the Noyantis CommandBars wrapper template is now available. This is an Interim release. It contains the following updates: Codejock 13.4.2 -> 15.0.2 compatibility added; 'Accepted' Embed point generated for each Popup option specified on a control; New method: 'DeleteAllComboOptions' ; New method: 'DeleteControlPopupItems' ; Two new methods added to help with other 3rd Party compatibility. The update is free to all users who have an Active Maintenance Plan in place, or costs $47.50 for all users who have a Lapsed Maintenance Plan. It can be downloaded via the Products page within the members area of the web site.

Posted April 25 2011 (permanent link)

CLARION MAGAZINE
READ. LEARN. SOLVE.

Home | Subscribe | E-Books | News | Blog | Store | My ClarionMag | My Lists | Contact

The ClarionMag Blog

April articles still to come

April's articles have been piling up on the editor's desk, and will be appearing on the site shortly. Here are some of the things you can look forward to this month:

- S Jayashanker shows a clever trick for filtering hand coded list boxes.
- Steve Parker explains how to get rid of __Dont_Touch_Me__ INI files.
- Steve also takes another step on the dark road to OOP. Is it devolution, or is it Steve-O?
- While not a month owned by Dr. Parker, there does seem to be a theme developing. John Morter goes on a distinctly Parker-esque journey of discovery, unraveling the tangled web that is browse refreshing.
- More tips and MagGems! (I won't be doing a ClarionLive MagGem this week, however - Good Friday is a holiday in Canada and I'll be away from the computer most of the day.)

And please take note of ClarionMag's spring subscription sale! If you're not sure of the status of your subscription just go to the My ClarionMag page.

Posted April 20 2011 (permanent link)

Z at ClarionLive Tomorrow

Bob Z will be on the ClarionLive Webinar tomorrow for the first 15 minutes for some special announcements. Remember, seating is limited to the first 100 participants. And Gordon Holfelder is back with a presentation on dashboards.

Posted April 28 2011 (permanent link)

C8 review under way

I'm working on a "first look" review of Clarion 8, and hope to have something posted this weekend.

At first glance, C8 seems a bit light on major new features - you notice mostly visual improvements, new locators, stuff like that. But there are a ton of smaller changes, and after a while they start to really add up. There's a lot to like in C8. Stay tuned.

Posted April 30 2011 (permanent link)

Tip of the Week: Forcing Code Completion

By Dave Harms

Posted April 7 2011

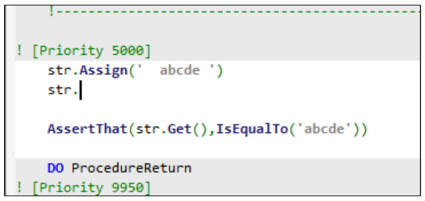
Clarion 7's code completion isn't what I'd call utterly reliable, but it works well enough that I've become completely dependent on it, as I have in other development environments such as Visual Studio.

I realize that real programmers wouldn't be caught dead using code completion; they either know every required function, variable, method and class name by heart, or they've got a stack of books on the shelf by people like Peter Norton and they know how to use them.

Me, I'm spoiled. I like being coached by the IDE. But code completion doesn't always work when you want it to work; sometimes it needs a little coaching before that window pops up.

There are two things I find myself doing with regularity to invoke code completion.

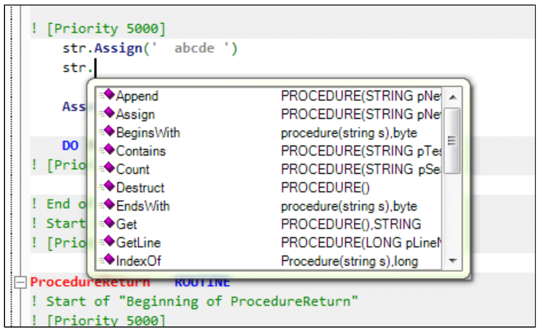
First, whenever I'm working with groups, queues, files or classes, I often press backspace followed by a period. In Figure 1 `str` is an instance of the `CM_System_String` class, and although the cursor is right after the period no code completion window is displayed. Most likely that's because I've just moved my cursor to that position; usually it takes a period to make the code completion window pop up. So....



```
    !--------------------------------------------
! [Priority 5000]
    str.Assign('  abcde ')
    str.

    AssertThat(str.Get(),IsEqualTo('abcde'))

    DO ProcedureReturn
! [Priority 9950]
```

Figure 1. No code completion displayed

In Figure 2 I've pressed backspace followed by a period, and presto! There's the code completion window.



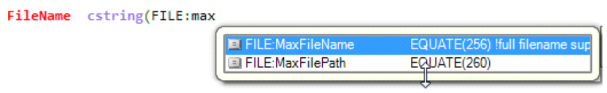
```
! [Priority 5000]
    str.Assign('  abcde ')
    str.

    Ass

    DO
! [Prio

! End o
! Start
! [Prio

ProcedureReturn    ROUTINE
! Start of "Beginning of ProcedureReturn"
! [Priority 5000]
```

Append	PROCEDURE(STRING pNe
Assign	PROCEDURE(STRING pNe
BeginsWith	procedure(string s),byte
Contains	PROCEDURE(STRING pTe
Count	PROCEDURE(STRING pSe
Destruct	PROCEDURE()
EndsWith	procedure(string s),byte
Get	PROCEDURE(),STRING
GetLine	PROCEDURE(LONG pLineN
IndexOf	Procedure(string s),long

Figure 2. Forcing the code completion window to appear

This is an old habit of mine, but in fact there's an even easier way to bring up the code completion window: just press Ctrl-Spacebar. If your cursor is right after the period, as in Figure 2, Ctrl-Spacebar will also bring up the code completion window.

I do regularly use Ctrl-Spacebar when I'm halfway through typing something I think I know the name of, but I'm not completely sure. In Figure 3 I've started to declare a cstring that I want to be as big as the maximum allowed filename. I can never remember that equate, for some reason. And if code completion for whatever reason isn't offering, pressing Ctrl-Spacebar usually does the trick.



```
FileName  cstring(FILE:max
```

FILE:MaxFileName	EQUATE(256) !full filename sup
FILE:MaxFilePath	EQUATE(260)

Figure 3. Pressing Ctrl-Spacebar

Note that in Figure 3 the code completion window is filtered to just those items that begin with the text I've already typed.

Sometimes code completion just doesn't work, for whatever reason. But I find these two tricks get me by in most situations where the code completion window hasn't, for whatever reason, popped up.

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

Tip of the Week: Window Previewer Tricks

By Dave Harms

Posted April 15 2011

Clarion has had a window previewer since, well, I can't remember when. It's a little piece of the IDE that takes your window structure, compiles it, and displays it so you can see exactly how it will look when it's running in your program.

Initially C7 didn't have a window previewer. The official line was that one wasn't necessary, but enough users disagreed (some quite strongly - you know who you are) and the previewer was fairly quickly added back into the product.

There are two ways to launch the previewer. One is to click on the previewer icon (Figure 1); the other is to press Ctrl-Shift-V.

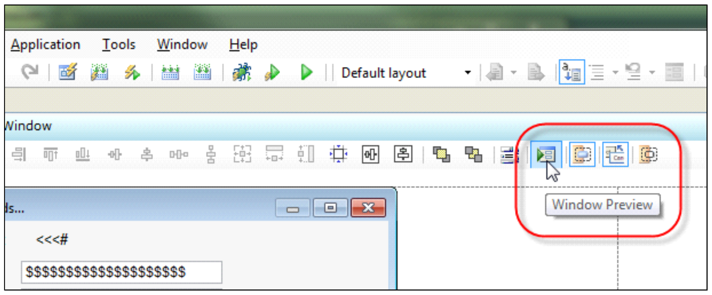
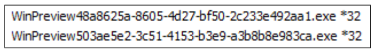


Figure 1. The previewer icon

The C7 window previewer compiles your window structure and runs it as a *separate program*. You can see these instances in Task Manager (Figure 2).



```
WinPreview48a8625a-8605-4d27-bf50-2c233e492aa1.exe *32
WinPreview503ae5e2-3c51-4153-b3e9-a3b8b8e983ca.exe *32
```

Figure 2. Window previewer instances

This is pretty handy for comparing window versions. As noted in a SoftVelocity blog post from 2009, you can launch a previewer, make changes to your window (or enable/disable visual styles), and then compare your new window side by side with your original window.

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

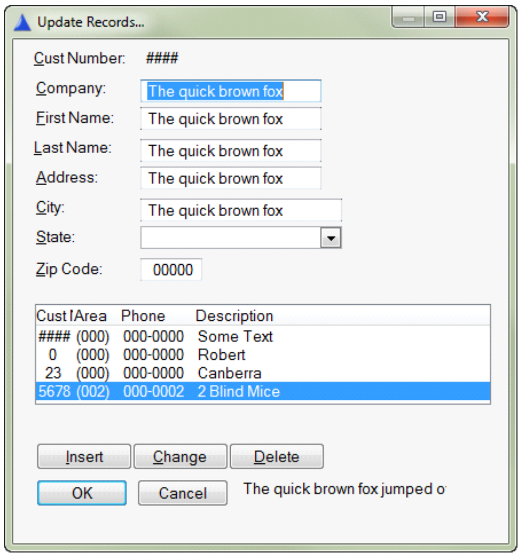
BACK TO TOP

Tip of the Week: Customizing the Window Previewer's Data

By Dave Harms

Posted April 22 2011

Last week's tip was about running multiple window previewer instances. If you've tried the previewer at all, one of the things you probably noticed about the C7 previewer versus the C6 previewer is the newer version supplies dummy data for display (Figure 1).


Update Records...
Cust Number: ####
Company: The quick brown fox
First Name: The quick brown fox
Last Name: The quick brown fox
Address: The quick brown fox
City: The quick brown fox
State:
Zip Code: 00000
Insert
Change
Delete
OK
Cancel
The quick brown fox jumped o

Cust I	Area	Phone	Description
####	(000)	000-0000	Some Text
0	(000)	000-0000	Robert
23	(000)	000-0000	Canberra
5678	(002)	000-0002	2 Blind Mice

Figure 1. Preview data

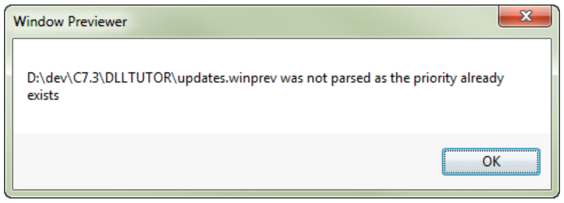
The preview data is controlled by a file called SoftVelocity.winprev, which you can find under your Clarion 7 install directory in the `data\Window Previewer` subdirectory. The Clarion Help provides information on the format of this file, noting among other things that you can create an application-specific version of the file called `appname.winprev` in your application directory.

The one critical thing you need to do when creating your custom previewer file is to change the priority from "0" to some other number:

```
<windowPreviewer
 name        = "Default controls"
 author      = "SoftVelocity Inc."
 url         = "www.softvelocity.com"
 priority    = "1"
 description = "Default values displayed for controls in the Window Previewer"
```

```
>
```

If you don't do this, the IDE will complain about conflicting priorities (Figure 2).


Window Previewer

D:\dev\C7.3\DLLTUTOR\updates.winprev was not parsed as the priority already exists

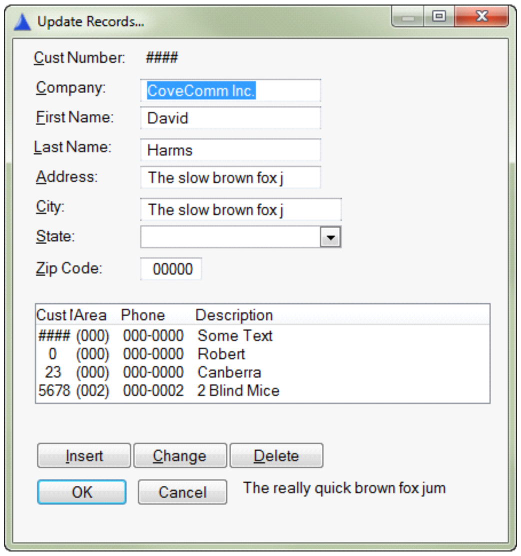
OK

Figure 2. Conflicting winprev priority

Whether or not you use an app-specific winprev file, you can create whatever preview data you like, including field-specific data. The following snippet overrides entry fields specific use variables:

```
<control type="Entry" use="FirstName">
     <text value="David"/>
</control>
<control type="Entry" use="LastName">
    <text value="Harms"/>
</control>
<control type="Entry" use="Company">
    <text value="CoveComm Inc."/>
</control>
```

Figure 3 shows the result.


Update Records...

Cust Number: ####
Company: CoveComm Inc.
First Name: David
Last Name: Harms
Address: The slow brown fox j
City: The slow brown fox j
State:
Zip Code: 00000

```
Cust IArea  Phone     Description
#### (000)  000-0000  Some Text
  0  (000)  000-0000  Robert
 23  (000)  000-0000  Canberra
5678 (002)  000-0002  2 Blind Mice
```

Insert | Change | Delete
OK | Cancel
The really quick brown fox jum

Figure 3. Customized display data

If you do make changes directly to the default winprev file, it's probably a good idea to take a backup before you upgrade to a later version of Clarion. The documentation also indicates you can put winprev files in userdata, but I haven't tried that option yet.

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

Getting Rid Of "[__Dont_Touch_Me__]" INI Files

By Steven Parker

Posted April 24 2011

When you create a new application, Clarion assumes you want an INI file.

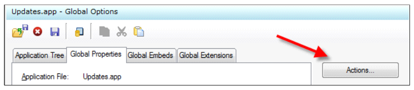
If you do not elect to save/restore window positions, you end up with an INI file whether you want one or not. Even if you don't use it, it will be there and it will contain the line:

```
[__Dont_Touch_Me__]
```

If you do not use the "Preserve" option (see Figure 2, below) and you do not use Save/Restore window position (there is a checkbox in Figure 2 to turn this off), you still end up with an INI file.

But you don't need or want it!

Start with Global Properties. Click the Actions button.



Updates.app - Global Options

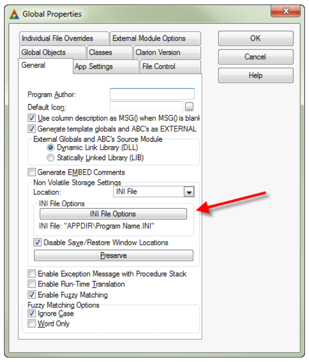
Application Tree | Global Properties | Global Embeds | Global Extensions

Application File: Updates.app

Actions...

Figure 1. Global Properties

On the Global Properties worksheet (pretty much unchanged for several generations of Clarion), click the INI File Options button.



Global Properties

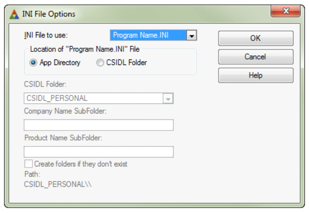
Individual File Overrides | External Module Options | Global Objects | Classes | Clarion Version | General | App Settings | File Control

OK | Cancel | Help

Program Author:
Default Icon:
[✓] Use column description as MSG() when MSG() is blank
[✓] Generate template globals and ABC's as EXTERNAL
External Globals and ABC's Source Module
(●) Dynamic Link Library (DLL)
( ) Statically Linked Library (LIB)
[ ] Generate EMBED Comments
Non Volatile Storage Settings
Location: INI File
INI File Options
INI File Options
INI File: "APPDIR\Program Name.INI"
[✓] Disable Save/Restore Window Locations
Preserve
[ ] Enable Exception Message with Procedure Stack
[ ] Enable Run-Time Translation
[✓] Enable Fuzzy Matching
Fuzzy Matching Options
[✓] Ignore Case
[ ] Word Only

Figure 2. INI File Options

Change the default INI File to use from the default values...



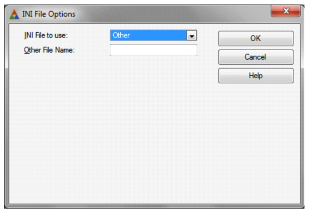
INI File Options

INI File to use: Program Name.INI
Location of "Program Name.INI" File
(●) App Directory ( ) CSIDL Folder
CSIDL Folder:
CSIDL_PERSONAL
Company Name SubFolder:
Product Name SubFolder:
[ ] Create folders if they don't exist
Path:
CSIDL_PERSONAL\\

OK | Cancel | Help

Figure 3. Clarion INI file defaults

... to use "Other:"



INI File Options

INI File to use: Other
Other File Name:

OK | Cancel | Help

Figure 4. Custom INI file

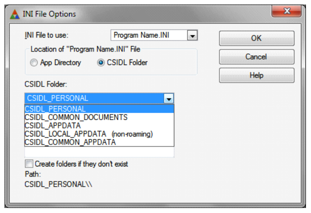
And leave the actual file name (second prompt) blank.

No INI file will be created. And, even better, no compiler or run time errors.

If you do need an INI file (which you will if you want to save/restore window positions), you have a number of options.

You can nominate your own file name, as in Figure 3. You can nominate your own file and folder, as in

Figure 4, by including the path in the “Other File Name” prompt (variables, with a leading exclamation point, are allowed). Or you can choose amongst a number of UAC-aware locations.



INI File to use: Program Name.INI

Location of "Program Name.INI" File: ( ) App Directory [✓ CSIDL Folder]

CSIDL Folder: CSIDL_PERSONAL

- CSIDL_PERSONAL
- CSIDL_COMMON_DOCUMENTS
- CSIDL_APPDATA
- CSIDL_LOCAL_APPDATA (non-roaming)
- CSIDL_COMMON_APPDATA

[ ] Create folders if they don't exist

Path:
CSIDL_PERSONAL\\

Figure 5. UAC-aware INI file locations

I believe these options were not working in some earlier versions of Clarion 7. They do work now. These options allow, for example, user-specific window sizes and locations, by choosing the personal folder. Of course, in that case, if you have global information (say, a SQL connection string, stored in an INI file, you may want to consider manually saving/retrieving that information as it is not shared if in the user’s folder).

But, the important part is: If you don’t want that “`[__Dont_Touch_Me__]` INI file,” you don’t have to have one.

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

Article comments

by Greg Miller on April 25 2011 (comment link)

`__Dont_Touch_Me__` can also be solved by making a small edit to ABUTIL.CLW. It's at line 901 of ABUTIL:IniClass.UpdateQueue in the 7.3.8222 release.

Change

```
SELF.Update (Sector, Name, RECORDS(Q))
```

to

```
IF RECORDS(Q)
  SELF.Update (Sector, Name, RECORDS(Q))
END
```

Then the section will only be created when it is needed.

by Dave Harms on April 25 2011 (comment link)

Thanks Greg, good tip! And remember to make that change next time you upgrade Clarion...

by Koen Tjoa on April 26 2011 (comment link)

Indeed a good tip, it made me look a bit deeper.

Won’t also work by commenting out the line: `SELF.UpdateQueue ('__Dont_Touch_Me__','Sectors',SELF.Sectors,SELF.Sectors.Family,SELF.Sectors.Item,SELF.Sectors.Type)` This line is in INIClass.Kill method (line 514 in Clarion 6.3 build 9058).

And I guess it can harm either to comment out the line in the INIClass.Init method (line 504 in Clarion 6.3 build 9058): `SELF.FetchQueue ('__Dont_Touch_Me__','Sectors',SELF.Sectors,SELF.Sectors.Family,SELF.Sectors.Item,SELF.Sectors.Type)`

Or am I overseeing something?

by Zeljko Manjkas on April 26 2011 (comment link)

For Clarion 6.3 instead comment in Kill method I used this code

IF NOT SELF.Sectors &amp;= NULL

IF RECORDS(SELF.Sectors) ! My line

```
 SELF.UpdateQueue('__Dont_Touch_Me__','Sectors',SELF.Sectors,SELF.Sectors.Family,SELF.Sectors.Item,SELF.Sectors.T
```

END ! My line

DISPOSE(SELF.Sectors)

SELF.Sectors &amp;= NULL

END

Comment in init method is not neaded.

BACK TO TOP

Filtering Hand Coded Lists Using The TREE Attribute

By S Jayashankar

Posted April 25 2011

I recently had to do some financial reports for a customer of mine which required a drill down effect. As a result of that work I discovered that I could use the tree attribute to filter rows from the report.

Since my customer's reports needed to be generated from a pre-defined setup file which decided the different levels of summarization for the report, I opted to do the report in a List Box with a Tree View that could be printed or exported later (using EasyExcel). I initially tried out the In-Memory driver to hold the data and the Standard Browse template to display it, but the scroll bar did not behave properly for a file loaded Browse with a Tree column. So I opted to hand code the tree view; I used a queue for the data which was linked to a List Box control.

I also wanted to provide an easy way of expanding/contracting the levels in the tree view without the user having to manually do the same. To do this, I created a local variable to hold the Display level. When the user changed the level, I ran through the queue and contracted all lower levels. In a tree view, rows with a positive level number are shown, those with a negative level are hidden. If the Display Level was 2, I set each row level with a value of 1 as +1 (expanded) and set all row levels greater than 1 as –(row level) (contracted).

As I was doing this, it occurred to me that I could use the Level to filter out rows from a List Box. Also, when you set the Tree attribute for a List Box column you have set an additional option to disable the display of the Level hierarchy i.e. indentation with boxes & lines instead of + and - icons. This would put the expanded/contracted state of the tree under the programmer’s control only. In the past, whenever I needed to filter the rows of a hand coded List Box, I always resorted to having a data queue and a display queue i.e. the data queue would hold all the data and the display queue would hold only the records to be displayed and would be linked to the List Box. I was pretty excited about this technique, and even though I knew it would work (as it was working for my reports) I decided to write a small program to test it with normal data.

I started with a simple queue containing First Name, Last Name & Company with the First Name column designated as the tree column. I added an extra `LONG` column after the First Name to hold the row level:

```
TreeQ            QUEUE,PRE(TQ)
FirstName            STRING(30)
FirstName_Level      LONG
LastName             STRING(30)
Company              STRING(40)
```

```
                END
```

For the filtering, I created local variables for the First Name, Last Name & Company. I also added a Filter Type so I could decide whether the filtering should be based on a “Starts With”, “Contains” or an “Exact Match”.

```
 SEL:FilterType  STRING(1)   ! S - Starts With, C - Contains, E - Exact Match

 SEL:FirstName   STRING(30)
 SEL:LastName    STRING(30)
 SEL:Company     STRING(40)
```

After I used the Window Designer (Ctrl-D) to populate the controls, my formatted window looked like Figure 1.

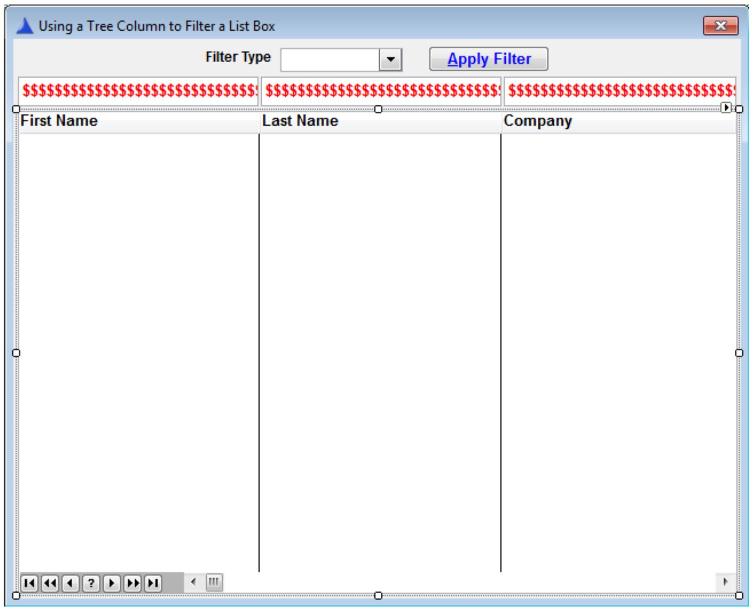
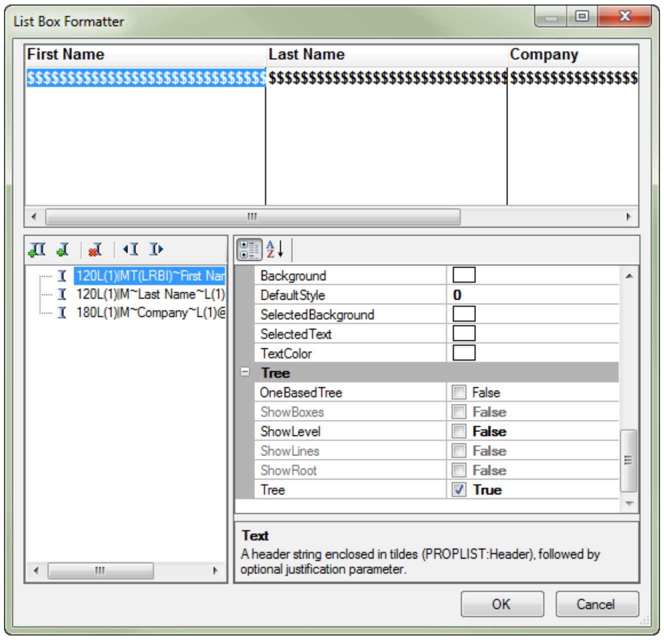


Figure 1. The formatted window

There is nothing special to note about the properties of the controls except for the First Name column of the List Box (shown below). Notice that the Tree option has been checked (set to True) to make this column a tree column and the ShowLevel option has been unchecked (set to False) to suppress the display of the boxes & lines normally associated with a tree display (Figure 2).


List Box Formatter

First Name	Last Name	Company
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$	$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$	$$$$$$$$$$$$$$$$

120L(1)|MT(LRBI)~First Nar
120L(1)|M~Last Name~L(1)
180L(1)|M~Company~L(1)@

Background	
DefaultStyle	0
SelectedBackground	
SelectedText	
TextColor	
Tree	
OneBasedTree	False
ShowBoxes	False
ShowLevel	False
ShowLines	False
ShowRoot	False
Tree	✓ True

Text
A header string enclosed in tildes (PROPLIST:Header), followed by optional justification parameter.

Figure 2. Tree options

During the initial population of the Tree queue (`TreeQ`) with data, I set the row level (`TQ:FirstName_Level`) of all rows to -1. When the user enters a filter string and presses the “Apply Filter” button, I call a routine which updates the row level column of all the rows. For a row to be suppressed, it needs the previous row to be contracted (negative row level) and it should have a row level with a greater absolute value than the previous level.

Since I had set all the row levels to -1 (to allow for following rows to be hidden), the simple answer was to set the row level as -2 for all those rows which should not be displayed (filtered). This worked fine except when the first row needed to be filtered (as there was no preceding row for it). My first solution was to add a blank row (note the commented code in the attached source) which worked fine except for the unwanted display of an extra line on top. Also, this extra blank line would require the programmer to code around this blank row if the program was executing code when the user clicked on a row.

After a good night’s sleep, the solution presented itself – just re-sort the queue to make sure that the row levels of -1 remained on top and the row levels of -2 were at the bottom of the queue. To achieve this, I sort the queue as row level (Descending order) followed by all the standard sort columns in whatever order they needed to be. The code below is the code used to filter and sort the queue:

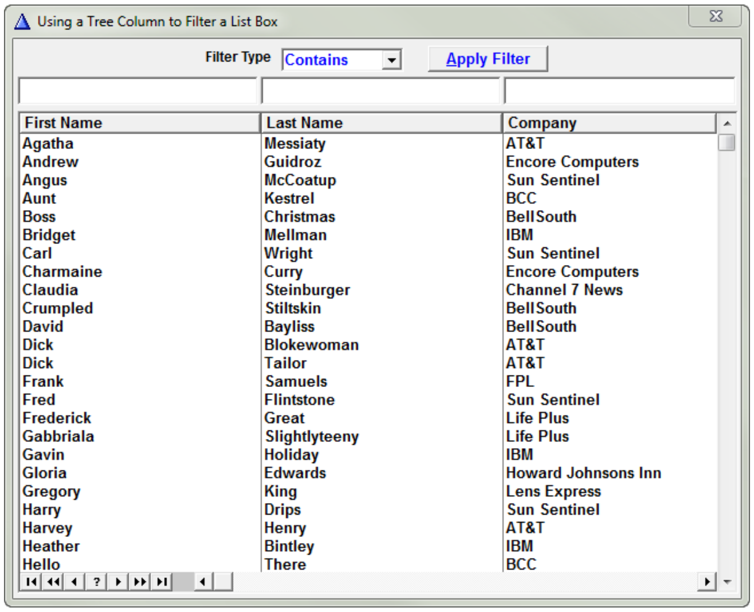
```
FilterList                                ROUTINE

    Len_FN# = LEN(CLIP(SEL:FirstName))
    Len_LN# = LEN(CLIP(SEL:LastName))
    Len_CO# = LEN(CLIP(SEL:Company))

    IF AddBlankLineOrSort
        SORT(TreeQ, +TQ:FirstName, +TQ:LastName, +TQ:Company)

        !GET(TreeQ, 1)
        !DELETE(TreeQ)
```

```
END

AddBlankLineOrSort = False

LOOP R# = 1 TO RECORDS(TreeQ)
    GET(TreeQ, R#)

    TQ:FirstName_Level = -1

    IF Len_FN#
        CASE SEL:FilterType
        OF 'C'
            IF NOT INSTRING(UPPER(SEL:FirstName[1 : Len_FN#]), |
                UPPER(TQ:FirstName), 1, 1)
                THEN TQ:FirstName_Level = -2.
        OF 'S'
            IF UPPER(TQ:FirstName[1 : Len_FN#]) <> |
                UPPER(SEL:FirstName[1 : Len_FN#])
                THEN TQ:FirstName_Level = -2.
        OF 'E'
            IF UPPER(TQ:FirstName) <> UPPER(SEL:FirstName)
                THEN TQ:FirstName_Level = -2.
        END
    END

    IF Len_LN#
        CASE SEL:FilterType
        OF 'C'
            IF NOT INSTRING(UPPER(SEL:LastName[1 : Len_LN#]), |
                UPPER(TQ:LastName), 1, 1)
                THEN TQ:FirstName_Level = -2.
        OF 'S'
            IF UPPER(TQ:LastName[1 : Len_LN#]) <> |
                UPPER(SEL:LastName[1 : Len_LN#])
                THEN TQ:FirstName_Level = -2.
        OF 'E'
            IF UPPER(TQ:LastName) <> UPPER(SEL:LastName)
                THEN TQ:FirstName_Level = -2.
        END
    END

    IF Len_CO#
        CASE SEL:FilterType
        OF 'C'
            IF NOT INSTRING(UPPER(SEL:Company[1 : Len_CO#]), |
                UPPER(TQ:Company), 1, 1)
                THEN TQ:FirstName_Level = -2.
        OF 'S'
            IF UPPER(TQ:Company[1 : Len_CO#]) <> |
                UPPER(SEL:Company[1 : Len_CO#])
                THEN TQ:FirstName_Level = -2.
        OF 'E'
            IF UPPER(TQ:Company) <> UPPER(SEL:Company)
                THEN TQ:FirstName_Level = -2.
        END
    END

    PUT(TreeQ)

    IF R# = 1 AND TQ:FirstName_Level = -2
        THEN AddBlankLineOrSort = True.
END

IF AddBlankLineOrSort
    !CLEAR(TreeQ) ; TQ:FirstName_Level = -1 ; ADD(TreeQ, 1)
    SORT(TreeQ, -TQ:FirstName_Level, +TQ:FirstName, |
        +TQ:LastName, +TQ:Company)
END

DISPLAY(?List:TreeQ)

EXIT
```

Since the queue was initially sorted as (`+TG:FirstName, +TG:LastName, +TG:Company`), I needed to re-sort the queue as (`-TG:FirstName_Level, +TG:FirstName, +TG:LastName, +TG:Company`).

That’s it.

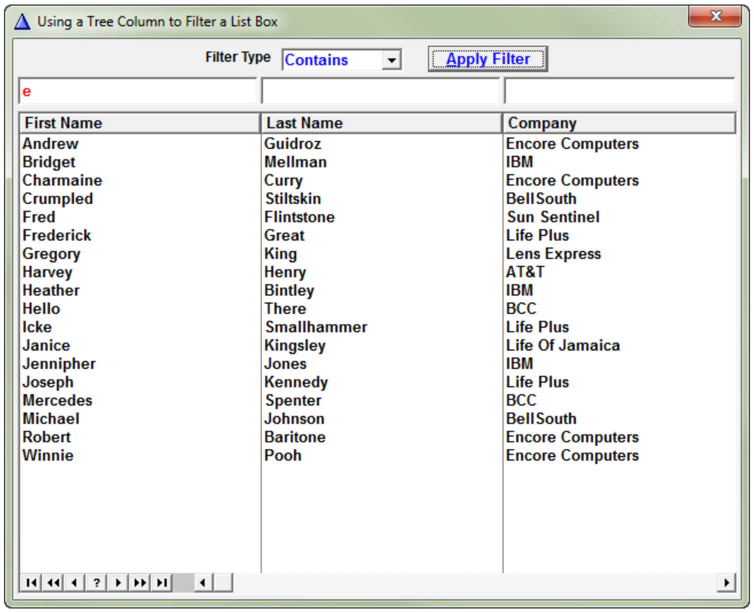
Figure 3 shows the sample app with no filtering.



First Name	Last Name	Company
Agatha	Messiaty	AT&T
Andrew	Guidroz	Encore Computers
Angus	McCoatup	Sun Sentinel
Aunt	Kestrel	BCC
Boss	Christmas	BellSouth
Bridget	Mellman	IBM
Carl	Wright	Sun Sentinel
Charmaine	Curry	Encore Computers
Claudia	Steinburger	Channel 7 News
Crumpled	Stiltskin	BellSouth
David	Bayliss	BellSouth
Dick	Blokewoman	AT&T
Dick	Tailor	AT&T
Frank	Samuels	FPL
Fred	Flintstone	Sun Sentinel
Frederick	Great	Life Plus
Gabbriala	Slightlyteeny	Life Plus
Gavin	Holiday	IBM
Gloria	Edwards	Howard Johnsons Inn
Gregory	King	Lens Express
Harry	Drips	Sun Sentinel
Harvey	Henry	AT&T
Heather	Bintley	IBM
Hello	There	BCC

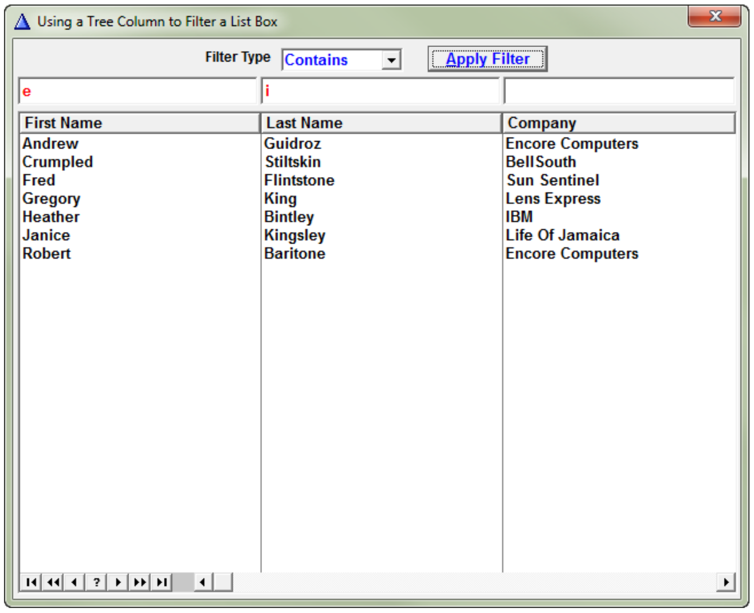
Figure 3. The sample app

In Figures 4 through 6 I've progressively added single character "contains" filters.



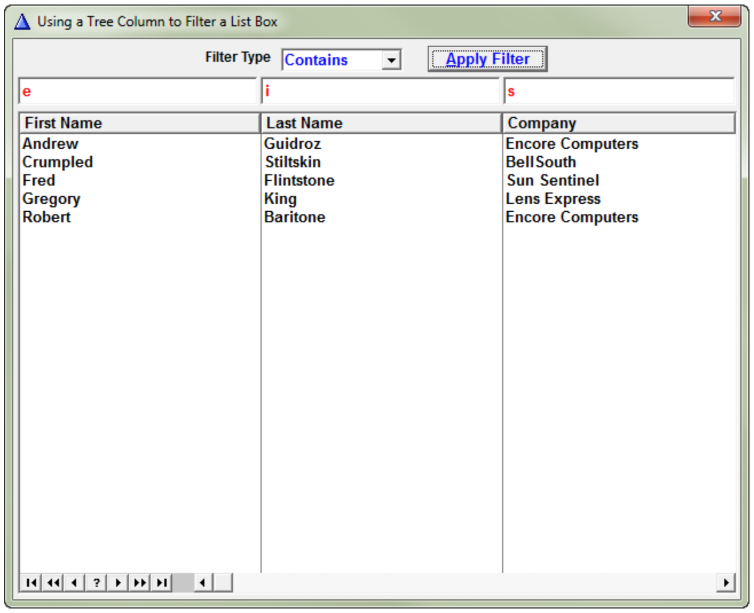
First Name	Last Name	Company
Andrew	Guidroz	Encore Computers
Bridget	Mellman	IBM
Charmaine	Curry	Encore Computers
Crumpled	Stiltskin	BellSouth
Fred	Flintstone	Sun Sentinel
Frederick	Great	Life Plus
Gregory	King	Lens Express
Harvey	Henry	AT&T
Heather	Bintley	IBM
Hello	There	BCC
Icke	Smallhammer	Life Plus
Janice	Kingsley	Life Of Jamaica
Jennipher	Jones	IBM
Joseph	Kennedy	Life Plus
Mercedes	Spenter	BCC
Michael	Johnson	BellSouth
Robert	Baritone	Encore Computers
Winnie	Pooh	Encore Computers

Figure 4. Filtering by First Name



e	i	
First Name	Last Name	Company
Andrew	Guidroz	Encore Computers
Crumpled	Stiltskin	BellSouth
Fred	Flintstone	Sun Sentinel
Gregory	King	Lens Express
Heather	Bintley	IBM
Janice	Kingsley	Life Of Jamaica
Robert	Baritone	Encore Computers

Figure 5. Adding a Last Name filter



e	i	s
First Name	Last Name	Company
Andrew	Guidroz	Encore Computers
Crumpled	Stiltskin	BellSouth
Fred	Flintstone	Sun Sentinel
Gregory	King	Lens Express
Robert	Baritone	Encore Computers

Figure 6. Adding a Company filter

Download the source

Article comments

BACK TO TOP

ClarionMag's Spring Sale Ends Wednesday, April 27

Posted April 25 2011

Subscription sales have become a rarity at Clarion Magazine. But it's been a long and grueling winter up here on the Canadian Prairies, and with spring finally taking hold it seems like a good time to do some celebrating!

Until Wednesday, April 27 you can get a Silver Subscription, which includes all the back issues and a whole pile of other goodies, for **just $99**. You **save $26!** Or get a two year Silver Subscription for **only $188, and save $62** (compared to two regular $125 subscriptions).

You can even save a few bucks on our entry level Bronze subscription, just $55 during the sale.

Don't wait **- subscribe/renew now!** April 27 is just around the corner, after which we're back to our regular subscription rates.

BACK TO TOP

In Memory Of Nigel Hicks

Posted April 28 2011

Word comes from David Bayliss that Nigel Hicks has passed away. Nigel was found on Tuesday at his home in England, by his brother, and could not be revived.

Nigel was a co-founder of JPI and the Director of Product Development at the London Development Centre. He developed the original Clarion IDE framework and the Report Writer. He was also the development team leader when SoftVelocity acquired the Clarion product line. Nigel contributed greatly to Clarion’s success over the years.

On behalf of the Clarion community, I would like to express condolences to Nigel’s family and friends on their loss.

My thanks to Russ Eggen and Bob Foreman for passing along the news.

Article comments

by rkchapman on April 28 2011 (comment link)

Before I met Nigel, he had worked at Borland with Ole, Jesper, and Niels – he worked on Sidekick and the Turbo Pascal system. I think he was probably their first non-Danish employee. When Borland decided to acquire a C compiler rather than use the one that the team had been developing, the team broke away and set up JPI (Jensen and Partners International) to create a rival range of compilers. I first met Nigel when he interviewed me for a job at Jensen and Partners International (JPI) back in 1990. The first thing that struck me about him was how he was not afraid of any problem or restriction that a computer or operating system might present, he would always find a way to make the program work. Previously I had just assumed that you took what the operating system gave you, but Nigel was happy to intercept interrupts, write TSR (terminate and stay resident) programs etc (in fact he more or less invented the concept of TSR's with the original Borland sidekick program).

JPI ran into money problems and was acquired (for its technology and its developers) by one of its customers, Clarion Software. Clarion too ran into money problems and Nigel and the rest of the development team was again acquired by a customer, eData.com. eData (later renamed Seisint) was acquired (partially for the technology that Nigel had co-invented and co-developed) by LexisNexis in 2003.

While at Seisint and continuing after the acquisition by LexisNexis, Nigel and the team developed a supercomputing platform to allow thousands of computers to co-operate on data processing tasks. He is cited as the co-inventor on a dozen or so patents or pending patents relating to this technology. In particular he developed a new way of sorting data using the combined power of these thousands of machines, which enabled our technology to sort data at previously impossible speeds and/or to sort previously impossibly large datasets.

When I first worked at JPI, Nigel was my boss. Later he ended up reporting to me. I don’t recall exactly when or how it happened– that’s how insignificant the event was. I think he conned me into taking over so he could avoid dealing with all the management crap and focus on the programming that he loved. We were always colleagues and partners – the reporting structure was irrelevant.

Nigel could debug the most obscure of bugs from the flimsiest of evidence – give him a core dump or a log file and he’d be able to tell you exactly what a program had been doing in the vital moments before it failed. The TopSpeed debugger – which he wrote – was the first debugger I ever saw that combined high-level and low level debugging and was a joy to use compared to anything that had gone before. One of our customers – a security agency – was unable to provide us with log files or core dumps (they were classified) and could only describe their bugs in terms of “We ran a program – can’t tell you want it looked like or what it did – and it failed – can’t tell you exactly what the error message was. And Nigel was still able to work out the bugs, and fix them.

Nigel was always available to look at issues – 24 hours a day, 365 days a year, if you sent him an email or instant message, you’d get a reply within an hour. We used to joke that he must have an alarm clock wired to his computer to wake him if a message came in. More recently, with his illness, he would warn me that he wouldn’t be working as much, but it never seemed to slow him down detectably. He still worked harder and longer than anyone else I know.

Nigel was great at arguing – sometimes frustratingly so – and not always good at compromise. But we always ended up with solutions that were better than we would have got if he had not argued his point of view so strongly.

Nigel had been unwell for some time, but his passing still came as a great shock to us all. We will all miss him enormously.

---

by Dave Harms on April 28 2011 (comment link)

That's terrific, Richard. Thank you so much for giving us some insight into the man.

Dave

---

by Carl Sumner on May 2 2011 (comment link)

He will be missed.

I used JPI Modula-2 before I used Clarion. It is clear to see that one or more geniuses had their hands on both of them. Before JPI helped them, Clarion was a pre-compiled virtual language that ran somewhat slow. The difference after JPI worked on it was startling. We also have them to thank for the other compilers that are still part of Clarion. And, for the more efficient argument passing that makes use of cpu registers.

I never met him, but I wish I had...

Carl Sumner

---

by Paul Attryde on May 5 2011 (comment link)

Many years ago, when I was young and stupid and fresh out of college, one of the first real jobs I had was providing tech support for Jensen &amp; Partners (JPI). I sat on the mezzanine in the office in Clerkenwell, looking down on the development team and trying to resist the urge to throw dead 5 1/4 disks into the ceiling of the pitched roof above us.

Nigel had a tiny little office, filled with books and magazines and piles of paper and floppy disks, where he'd work on things like the debugger and the dos extender and other equally complicated things. There was never any doubt in anyone's mind that he was extremely clever, and he was always accommodating when we'd go down and ask questions about problems that customers were having. I was there for barely 6 months before Bruce bought JPI and I moved to the sales office in Harpenden to get intimately familier with CDD 300x patching. I rarely saw anyone from TSDC again.

Sitting here now, older and slightly less stupid, I can say without a shadow of a doubt that Nigel was one of the most intelligent people I've ever met. It was an honour to have worked with him.

BACK TO TOP

CLARION MAGAZINE
READ. LEARN. SOLVE.
Home Subscribe E-Books News Blog Store My ClarionMag My Lists Contact

How To Have Your [Browse] Cake And Eat It Too (a refreshing story)

By John Morter

Posted April 29 2011

I invite you to accompany me on a Dr. Parkeresque journey of discovery.

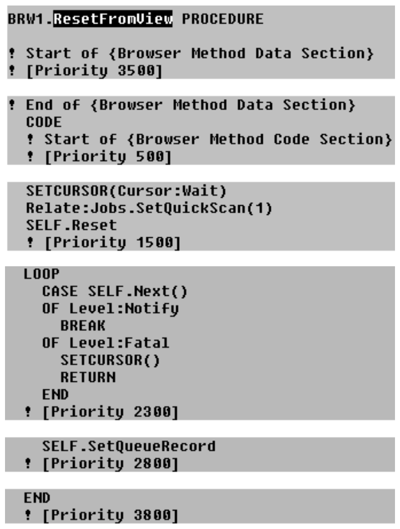
There will be a conundrum to start with, a number of dead-ends and surprise encounters along the way, and an enlightenment at the end to (hopefully) make the journey worth your while.

This journey started for me when I was exploring Vince Sorensen's excellent ABC Free template set. I came across one particular browse-related template, named `BrowseResetFromViewSkipLoopThruFileGlobal`, that started me wondering. The commentary provided when the template is applied claimed that:

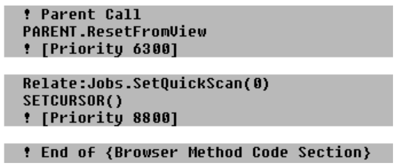
> This template causes the derived Browse.ResetFromView method to simply call the Parent and then return when the browse is page-loaded and has no totaling (resulting in faster browse loading).

Vince's template note caught my eye because I've had an inkling that my browses were sometimes being loaded twice. That is, it seemed to me (for reasons I could never quite work out) that the process of reading through source table(s) and populating the list/Queue could be `LOOP`ing through more times than the expected once-only. I figured I might learn something by digging a bit deeper.

Figure 1 shows the generated code before the template is applied.



```
BRW1.ResetFromView PROCEDURE

! Start of {Browser Method Data Section}
! [Priority 3500]

! End of {Browser Method Data Section}
  CODE
  ! Start of {Browser Method Code Section}
  ! [Priority 500]

  SETCURSOR(Cursor:Wait)
  Relate:Jobs.SetQuickScan(1)
  SELF.Reset
  ! [Priority 1500]

  LOOP
    CASE SELF.Next()
    OF Level:Notify
      BREAK
    OF Level:Fatal
      SETCURSOR()
      RETURN
    END
  ! [Priority 2300]

    SELF.SetQueueRecord
  ! [Priority 2800]

  END
  ! [Priority 3800]
```

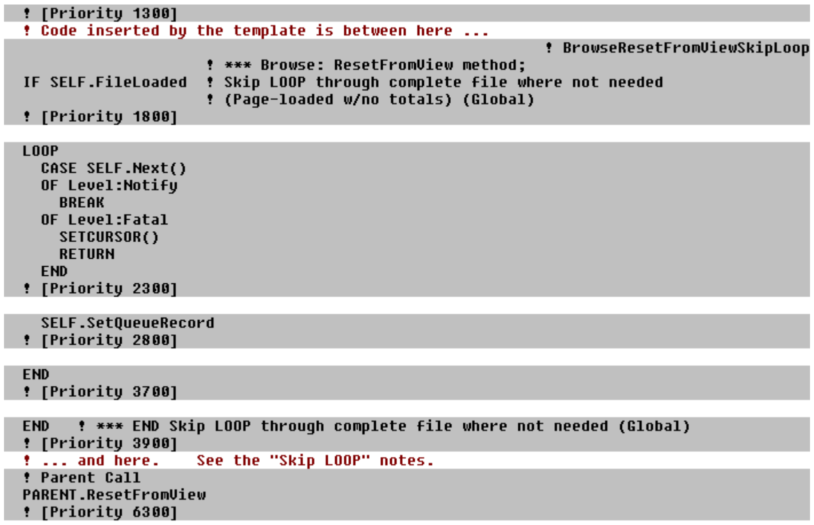


```
  ! Parent Call
  PARENT.ResetFromView
  ! [Priority 6300]

  Relate:Jobs.SetQuickScan(0)
  SETCURSOR()
  ! [Priority 8800]

  ! End of {Browser Method Code Section}
```

Figure 1. Standard (ABC) browse code, before the template was applied

> **Aside:** If you'1re not already taking advantage of Vince's excellent template set then I encourage you to take a look. The package contains a treasure-trove of templates of every type, from features that facilitate your programming task to enhancements that make your resulting solution look better and work better for users. It contains at least three templates that I use in every single application that I craft. And it's free!

So, I applied the `~SkipLoopThruFile~` template to an application containing a simple browse procedure (on a table named `Jobs`) and then I took a look at the code it generated.

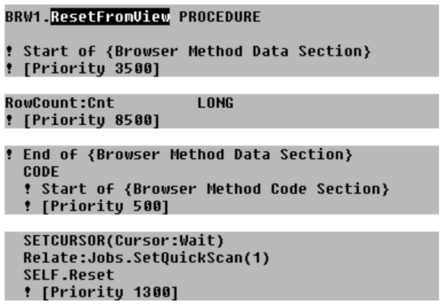


```
  ! [Priority 1300]
  ! Code inserted by the template is between here ...
                                                      ! BrowseResetFromViewSkipLoop
                            ! *** Browse: ResetFromView method;
  IF SELF.FileLoaded        ! Skip LOOP through complete file where not needed
                            ! (Page-loaded w/no totals) (Global)
  ! [Priority 1800]

  LOOP
    CASE SELF.Next()
    OF Level:Notify
      BREAK
    OF Level:Fatal
      SETCURSOR()
      RETURN
    END
  ! [Priority 2300]

    SELF.SetQueueRecord
  ! [Priority 2800]

  END
  ! [Priority 3700]

  END    ! *** END Skip LOOP through complete file where not needed (Global)
  ! [Priority 3900]
  ! ... and here.    See the "Skip LOOP" notes.
  ! Parent Call
  PARENT.ResetFromView
  ! [Priority 6300]
```

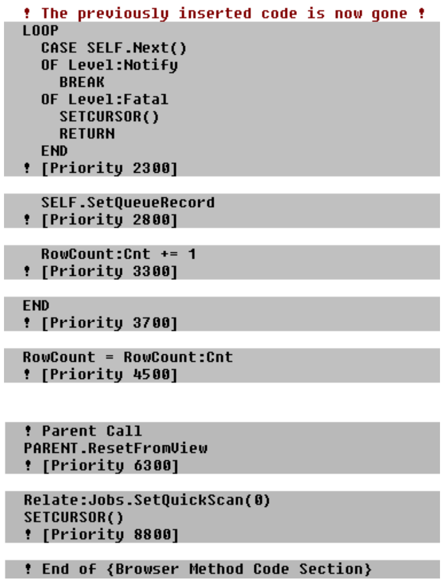
Figure 2. Same as Figure 1, after the template was applied

I found code inserted into the *browseclassname*`.ResetFromView` procedure. It's quite straightforward. IF the loading method prescribed for the browse (via the misspelled 'Default-Behavior' tab) is *not* set to file (loaded) then a section of code is completely skipped.

Mmmm! What happens, though, if the skipped `LOOP` is needed to accomplish totaling?



```
BRW1.ResetFromView PROCEDURE

! Start of {Browser Method Data Section}
! [Priority 3500]

RowCount:Cnt         LONG
! [Priority 8500]

! End of {Browser Method Data Section}
  CODE
  ! Start of {Browser Method Code Section}
  ! [Priority 500]

  SETCURSOR(Cursor:Wait)
  Relate:Jobs.SetQuickScan(1)
  SELF.Reset
  ! [Priority 1300]
```



```
! The previously inserted code is now gone !
LOOP
  CASE SELF.Next()
  OF Level:Notify
    BREAK
  OF Level:Fatal
    SETCURSOR()
    RETURN
  END
! [Priority 2300]

  SELF.SetQueueRecord
! [Priority 2800]

  RowCount:Cnt += 1
! [Priority 3300]

END
! [Priority 3700]

RowCount = RowCount:Cnt
! [Priority 4500]


! Parent Call
PARENT.ResetFromView
! [Priority 6300]

Relate:Jobs.SetQuickScan(0)
SETCURSOR()
! [Priority 8800]

! End of {Browser Method Code Section}
```

Figure 3. Unchanged, except that totaling of 'Each Record Read' is requested

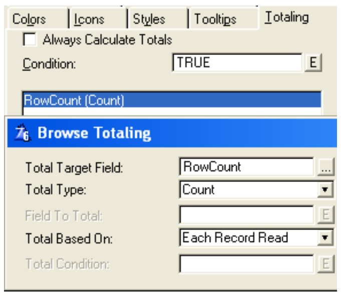
Aside: A peek at the ABC Free template code easily explains this 'disappearance'. The template is applied only on the condition that totaling has not been requested.

So, what can be deduced from this? There are a number of interesting inferences to explore.

1. It seems that *if* totaling has been requested then this `LOOP` is necessary, full stop.
2. The `LOOP` is also deemed necessary *if* totaling has *not* been requested *and* the list is file loaded. On the other hand, *if* totaling has *not* been requested *and* the list is page loaded then this LOOP (it seems) can be safely skipped.
3. The benefit of skipping the `LOOP` seems to be quite substantial, because it appears to be working its way through every record/row returned via the `View` (that's underlying the browse). Could this be the reason for the double loading (filling twice) of the `LIST` that I thought I'd sometimes observed? The fact that it's conditional (on me having requested totaling) would explain why I'd observed it only "sometimes". I was intrigued to understand more.
4. *This next one was not immediately apparent to me, but became so as I pondered*: The call to `SELF.SetQueueRecord` is executed only if totaling has been requested. Otherwise (it seems) this call is redundant. Mmmm!

Aside: The purpose of `SetQueueRecord` is to move items (that you have requested to appear in the browse list and/or have assigned as 'Hot Fields') from the table buffer into the Queue from which the browse list is displayed. It therefore contains the ideal embed point to do any 'massaging' of table data before it's populated into the Queue (*before* the `PARENT` call). However, even now at the time of writing this article, I cannot see the point of calling this procedure/Virtual for each row that's being totaled … Perhaps someone can enlighten me?

My next train-of-thought was to wonder what happens when the totaling is conditional, as is allowed for by the ABC Template implementation:



Always Calculate Totals
Condition: TRUE
RowCount (Count)

Browse Totaling

Total Target Field: RowCount
Total Type: Count
Field To Total:
Total Based On: Each Record Read
Total Condition:

Figure 4. Conditional totaling requested, on a count of 'Each Record Read'

The result confirmed that I'd need to keep digging – because now I was confused.

```
SETCURSOR(Cursor:Wait)
Relate:Jobs.SetQuickScan(1)
SELF.Reset
! [Priority 1300]
! Start of conditional totaling - from here ...
IF TRUE THEN
LOOP
  CASE SELF.Next()
  OF Level:Notify
    BREAK
  OF Level:Fatal
    SETCURSOR()
    RETURN
  END
! [Priority 2300]

  SELF.SetQueueRecord
! [Priority 2800]

  RowCount:Cnt += 1
! [Priority 3300]

END

! [Priority 3700]

RowCount = RowCount:Cnt
! [Priority 4500]

! Parent Call
PARENT.ResetFromView
! [Priority 6300]
! ... down to here
END !Conditional Totaling
Relate:Jobs.SetQuickScan(0)
SETCURSOR()
! [Priority 8800]
```

Figure 5. ABC Template code generated to execute the requested totaling

I was confused because, it seems, the call to `PARENT.ResetFromView` is not needed if the totaling condition is `FALSE`. Whereas, when totaling is unconditional this call is always performed (see Figure 3). And even when totaling has not been requested, this call is *still* always performed (see Figure 1). What the ?!

At this point, I decided I needed to be seeing what's really going on in the actual generated code, and not just what appears to be the case from within the Embeditor, as shown above. The best and easiest way to do this is via the Embeds/Filled button.

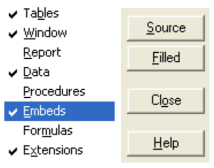


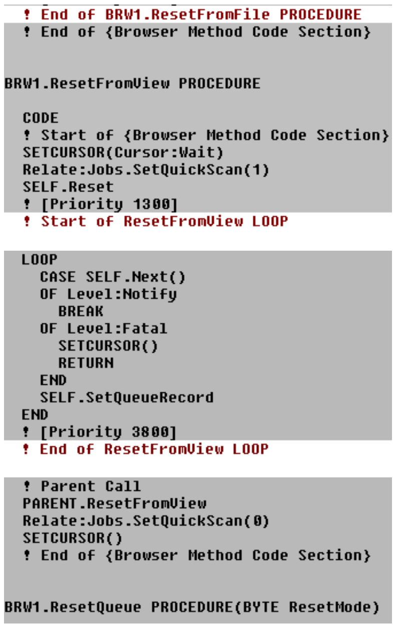
Figure 6. The Embeds/Filled button

Aside: The purpose of the Source button versus the Filled button is that the former shows all the code that could possibly be generated by the templates, with your embed code included within. This is the perspective to use when you're cruising to find the right/best embed point to achieve a specific tweak to the generated code.

The latter shows *only the code that will actually be generated by the templates*, with your embed code included within. This is the perspective to use when you want an uncluttered view of your embed code and its immediate surrounds.

Unfortunately, the last time I looked, not so long ago, this really useful feature was not working properly in Clarion 7. Yes, I did report the problem via Soft Velocity's PTSSystem (twice) but I guess I didn't explain it well enough, because I never received an answer. The lack of this feature is just one reason why I've not yet migrated to C7 … I'm now waiting for C8. (End of whinge, back to story).

I'm going to cut some corners here (for fear that I'll otherwise tire you out too much on the circuitous route I'm taking you on): I repeated various combinations of totaling, not totaling, conditional and unconditional … and checked the result each time via the Embed/Filled button.

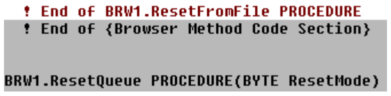


```
  ! End of BRW1.ResetFromFile PROCEDURE
  ! End of {Browser Method Code Section}


BRW1.ResetFromView PROCEDURE

  CODE
  ! Start of {Browser Method Code Section}
  SETCURSOR(Cursor:Wait)
  Relate:Jobs.SetQuickScan(1)
  SELF.Reset
  ! [Priority 1300]
  ! Start of ResetFromView LOOP

  LOOP
    CASE SELF.Next()
    OF Level:Notify
      BREAK
    OF Level:Fatal
      SETCURSOR()
      RETURN
    END
    SELF.SetQueueRecord
  END
  ! [Priority 3800]
  ! End of ResetFromView LOOP

  ! Parent Call
  PARENT.ResetFromView
  Relate:Jobs.SetQuickScan(0)
  SETCURSOR()
  ! End of {Browser Method Code Section}


BRW1.ResetQueue PROCEDURE(BYTE ResetMode)
```

Figure 7. The Embed/Filled equivalent of Figure 1 (with some comments)

The result was both surprising and worrying. It seems that this redundant `LOOP`ing code is generated even when I haven't requested totaling. Ah, so that's why I'd seen wasteful double loading of my browses!

Actually, no … I had jumped to a premature and incorrect conclusion. Here's what happens when those Start of LOOP/End of LOOP comments are removed.



```
  ! End of BRW1.ResetFromFile PROCEDURE
  ! End of {Browser Method Code Section}


BRW1.ResetQueue PROCEDURE(BYTE ResetMode)
```

Figure 8. The Embed/Filled equivalent of Figure 7 (with comments removed)

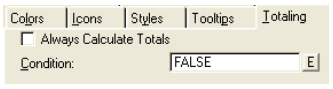
The penny finally dropped for me when I saw that the `ResetFromView` procedure had completely disappeared from the *actual* generated code (as a `Virtual`, overriding its `PARENT`). You may need to compare the top & bottom of figures 7 & 8 more closely, to see what I mean.

And a squiz at the ABC Template code confirmed my dawning suspicion; the code in ResetFromView is *conditionally* generated by the ABC Templates (see ABBROWSE.TPW) – it actually appears in the generated code *only* if totaling has been requested *or* (and this is the 'gotcha') if one has innocently dropped a few lines of embed code into the procedure, as I had.

OK – That explained a lot … but not everything.

My first reaction was to ensure, even if I were to innocently drop a line or two of embed code into this procedure sometime in the future, that my generated browse procedures wouldn't be handicapped by the completely unnecessary double loading of their lists' Queues.

My requirement for achieving this result is that I shouldn't need to remember to avoid the 'gotcha' … doing so needs to be automatic. Firstly, though, here's a simple, manual method of ensuring the redundant LOOPing code is always skipped: Set the Always Calculate Totals condition to False (Figure 9).



☐ Always Calculate Totals

Condition: FALSE

Figure 9. Conditional totaling set to IF FALSE, which is never True

Alternatively, the shipping ABC Template can be tweaked; this being my approach so that I don't have to remember to apply this setting for every new browse I ever generate. But, if you decide to follow my example, then do keep a backup of the unchanged template file (ABBROWSE.TPW) and be sure to document your changes … just in case.

Here's what my template code looks like, with line breaks added and my changes in bold:

```
#TAB('&Totaling'),HLP('~TPLControlBrowseBox_Totaling')
  #!DISPLAY
  #PROMPT('Always Calculate Totals',CHECK),%BrowseAllTotalOnOff
          ,DEFAULT(%False),AT(10)
  #ENABLE(NOT %BrowseAllTotalOnOff)
    #PROMPT('&Condition:',@S255),%BrowseAllTotalOnOffCondition
          ,REQ,AT(,,75),DEFAULT('False')
  #ENDENABLE
```

Now that I understood (a little) more, I decided to investigate the various combinations of page loading versus file loading, with and without totaling.

> **Aside:** Most of my day job development work is done in a SQL environment (of the Microsoft SQL Server persuasion). And I almost always choose to file load any browse that's based on a SQL table, because; i) it's wasteful of server side resources to be shuffling up-down partial result sets and passing small data packets across the LAN/WAN, and especially so in my case where the servers can be on the other side of the world, ii) I always contrive to force the user to make a table sub set selection so that only a reasonable number of records/rows are in the browse anyway, so there's no real disadvantage in file loading, and iii) many of the ABC template features, such as Filtered Locators, when applied to SQL tables, simply work better when the browse list is file loaded.
>
> When it comes to working with SQL tables, I'm a dedicated 'ABC purist' … more on this later.

For the time being though, all learnings and discoveries recounted here are backend bipartisan; they apply equally to SQL tables and TPS files.

Here's what I discovered, with reference to Figures 1 & 7:

1. If you embed any code into the *`BrowseClassName`*.`ResetFromView` procedure (even just a simple ! comment) then your browse list truly will be filled twice … on the assumption, by the ABC template, that you're doing totaling.

   Aside: This had me tricked for a bit because I had embedded a `STOP()` in the middle of the `LOOP` in a (misguided) attempt to see what was going on. Only to find, in a classic case of self delusion, that the `STOP()`, and therefore the `LOOP` too, was being executed. But, when I removed the `STOP()`, that it wasn't.

   The only good outcome of this distraction being I found the next point to be true.

2. At the point of entry into the `ResetFromView` procedure the browse list is *already* populated – but its vertical scroll bar has not yet been updated, or even enabled.

   This i) confirms the redundancy of *`BrowseClassName`*.`ResetFromView` *unless* totaling *is* required, and ii) shows there is a purpose to the call to `PARENT.ResetFromView` (either naturally or from within the *`BrowseClassName`* instance of `ResetFromView`), because this is what creates and updates the vertical scroll bar.

3. There is a minor bug (I humbly submit) in the ABC template code that implements conditional totaling. See Figure 5, and you'll notice that the call to `PARENT.ResetFromView` is located *within* the bounds of the condition. Which means that if the condition is never `TRUE` then `PARENT.ResetFromView` is never called – and, therefore, the scrollable thumb that you may be expecting is not implemented.

   Aside: It was only because I was looking at this so closely that I noticed this glitch. It's not something that you'd otherwise likely pick up on, because it occurs only *if* you have requested conditional totaling *and* your condition is never *true*, *and* you have requested a scrollable thumb … *else* you'll get a standard fixed thumb by default. All of which you'd need to be very diligent to spot.

   The fix (if you can be bothered, for such a minor blemish) is another tweak to the ABBROWSE.TPW file; Find the line containing: `END !Conditional Totaling` and change the number in the following `#PRIORITY()` statement to `7500`. Job done.

Totaling on a single pass

I then turned the direction of my ponderings to a bigger-picture question, "Is there any way I can have my cake and eat it too?" … or, "How can I get totaling, if I want it, without incurring the penalty of double-loading my browse list?".

There is actually a very easy way to get exactly this ideal result – but only if your aim is `COUNT`ing, not `SUM`ming or `AVG`ing … and only if you're file loading, not page loading, your browse. Here's how:

1) This tip leverages the fact that a browse list is based on a `QUEUE` – and, if the browse is file loaded

then the `QUEUE` will contain a full list of all rows that met the condition(s) applied by the `VIEW`'s filter. Therefore, a count of the records in the `QUEUE` will be a count of the rows presented within the browse.

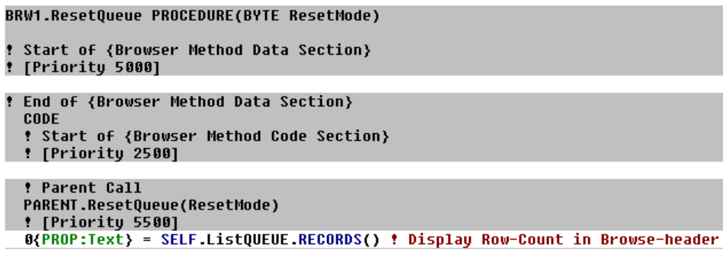
2) Either of the following equivalents will do the job for you; `VariableName = SELF.ListQUEUE.RECORDS()` - or - `VariableName = RECORDS(SELF.ListQueue)`.

However, unless you have made my suggested tweak to the ABBROWSE.TPW file (not the one just above, but the one documented via Figure 10), you should **not** plug one of these statements into your *`BrowseClassName`*`.ResetFromView`, because you'll be wasting your time … as I now realize I have probably been doing myself!

> **Note:** If that's not clear then you may have missed a key point above ... You'll be wasting your time because the template will detect your embed and will dutifully generate the `LOOP` construct into your to-be-compiled code – which will then double-load your browse list's `QUEUE` … thereby undoing the very aim of this tip.
>
> **Aside:** It's also quite possible for the penalty to be even greater than a duplicated read through the browse's source table. For example, due to the design of the SQL environment in which many of my apps are operating, my browses are commonly sourced from tables in one SQL database combined with description lookups from tables in other SQL database(s), not necessarily on the same SQL server.
>
> In this case, it's not wise to `PROJECT` all these logically-related tables into one `VIEW` structure, because, although it may possibly work (eventually), my users will have probably given up waiting and flicked my app a 'three fingered salute'.
>
> Instead, what I do is to define only the core table(s) in the `VIEW` and do explicit TryFetches for each 'lookup' table before the `PARENT` call in *`BrowseClassName`*`.SetQueueRecord`. (For an explanation of the purpose of this procedure, see my aside just before Figure 4, above.)
>
> The extra 'gotcha' here is due to the reference to this procedure from within the `LOOP` *`BrowseClassName`*`.ResetFromView`. So that, if this `LOOP` is triggered then it will both double load the browse list *and* wastefully repeat each of these lookups (potentially, in my case, via multiple servers located in different time zones).

Actually, there's an even more compelling reason not to embed this kind of code into the `ResetFromView` procedure – even though experience and instinct may suggest it's the correct and best place to embed code related to totaling – and this is because *if* you have specified a filtered locator for your browse you will not get the correct `COUNT` when the filtered locator is applied. That's because … well, I wish I knew; (I'll come back to you on this question. I now have the latest edition of The Editor's profiling tool set so I'll be able to track this down and find out, and I'll write a review on my findings).

Instead, the effective place to embed one of the equivalents above, where it will provide a correct `COUNT` result in all circumstances, including when a filtered locator has been applied, is *after* the `PARENT` call in the *`BrowseClassName`*`.ResetQueue` procedure.



```
BRW1.ResetQueue PROCEDURE(BYTE ResetMode)

! Start of {Browser Method Data Section}
! [Priority 5000]

! End of {Browser Method Data Section}
  CODE
  ! Start of {Browser Method Code Section}
  ! [Priority 2500]

  ! Parent Call
  PARENT.ResetQueue(ResetMode)
  ! [Priority 5500]
  0{PROP:Text} = SELF.ListQUEUE.RECORDS() ! Display Row-Count in Browse-header
```

Figure 11. How to have one's cake and get to eat it too

I could have happily halted at this point, because all my typical requirements were covered. That is to say, the only totaling I do is counting rows (I cannot recall when last I SUMmed or AVGed – though I can see that it might be useful - perhaps) and I'm usually working with SQL tables that I set to be file loaded. Therefore, I can pretty much always have my cake and eat it too, by completely avoiding the 'Totaling' tab and using my simple Figure 11 method instead.

SUM, AVG and COUNT on a single pass

But, I decided to set myself a challenge to see if I could find a way to achieve any of the 'Totaling' functions, on any field included in a browse-View, *without* reliance on double-loading the browse list's QUEUE … with knowledge in mind that SQL provides three basic aggregation functions that will handle my requirements: SUM(), AVG() and COUNT(). This led to an allowance that I granted myself; that my solution need only work in a SQL environment.

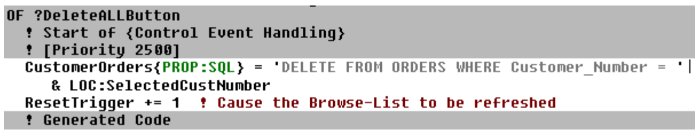
A SQL/ABC digression

A note here, regarding my Clarion development work in the SQL environment: I said, earlier, that when it comes to working with SQL tables I am a dedicated 'ABC purist'. What I meant by this is that I never commit the 'sin' of manually embedding "raw SQL" code directly into any of my Clarion apps.

I find this to be a 'sin' that's quite easy to avoid – because, in my experience, Clarion (from 6.2 on) does a truly excellent job of generating SQL statements, consisting of SELECT, WHERE and ORDER BY constructs, from the VIEWs upon which Clarion browse and Process procedures are based (including from table 'Keys', 'Record Filter' statements, 'Additional Sort Fields' and even 'Range Limits').

About the only time I find this not to be the case is where I need to do some type of bulk delete. For example, I may have a browse on a table of customers' orders with a filter by customer (where I force the user to select a customer before the browse will display anything at all). I have the usual Add, Change & Delete buttons that do the usual Clarion-thing … but I also want to provide an additional Delete ALL button that will delete all orders for the selected customer. In a SQL environment, it's *much* more efficient (in terms of both resource usage *and* speed) to do this all in one go, on the SQL Server back end, instead of issuing deletes row-by-row via the client app.

One method of achieving this might be to issue "raw SQL" code at the embed point where the Delete ALL button is ACCEPTed.



```
OF ?DeleteALLButton
  ! Start of {Control Event Handling}
  ! [Priority 2500]
  CustomerOrders{PROP:SQL} = 'DELETE FROM ORDERS WHERE Customer_Number = '|
     & LOC:SelectedCustNumber
  ResetTrigger += 1  ! Cause the Browse-List to be refreshed
  ! Generated Code
```

Figure 12. Committing the 'sin' of embedding "raw SQL"

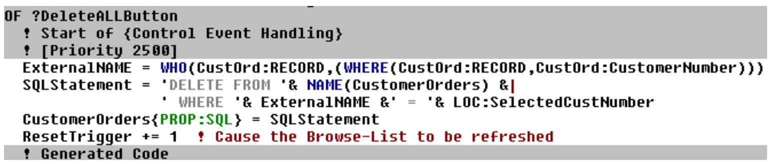
There are at least two reasons why I consider this to be a programming sin. Firstly, everything within the quoted string must be expressed in SQL syntax and it must refer to table names and column names exactly as they are defined (typically by a SQL DBA, not you or me) in the SQL database – and this can easily lead to confusion. For example, notice that the internal Clarion reference to the table known to SQL as “Orders” is “CustomerOrders”, and the Clarion Dictionary reference to the column known to SQL as “Orders.Customer_Number” is “CustOrd:CustomerNumber” (although you cannot see this in the snippet above).

It’s far too easy to get this wrong. And that would be acceptable, *I guess*, except for my second reason.

The Clarion compiler has no idea about any syntax mistakes that I may make within that string. Therefore, even on the optimistic assumption that my testing is totally thorough, I won’t find out about the mistake until SQL spits back a barely decipherable error to *me* – and it’s really tedious going around that loop. In the worst case, though, I may not know about it until a user calls the IT Help Desk; and she won’t have received an error message that’s nicely worded in user-speak!

Instead, my approach in this case is to expose all the potential ‘break points’ to the sturdy Clarion compiler. So that if I do make a blue in my coding then I’ll know about it well before the result gets anywhere near a punter.

The fail-safe alternative would look something like this;



```
OF ?DeleteALLButton
  ! Start of {Control Event Handling}
  ! [Priority 2500]
  ExternalNAME = WHO(CustOrd:RECORD,(WHERE(CustOrd:RECORD,CustOrd:CustomerNumber)))
  SQLStatement = 'DELETE FROM '& NAME(CustomerOrders) &|
                 ' WHERE '& ExternalNAME &' = '& LOC:SelectedCustNumber
  CustomerOrders{PROP:SQL} = SQLStatement
  ResetTrigger += 1  ! Cause the Browse-List to be refreshed
  ! Generated Code
```

Figure 13. Alternative ‘SQL direct’ coding, by a born-again ABC purist

Please note that, for clarity and simplicity purposes, I haven’t cluttered either of Figures 12 or 13 with the niceties of asking “OK?” before launching off on such a radical action, nor have I catered for any error checking. And, if you’re wondering about the ResetTrigger then see my article titled “Automagical Browse Refreshing”.

I sense some hecklers out there. One of them is saying; “Yeah, but Figure 12 is just one simple line of code – whereas Figure 13 takes three lines of gobbledygook”. To which I would reply; “True, but those three lines would not be represented many times at all in any of my apps (because the vast majority of my SQL statements are generated, transparently to me, via the SQL Driver) – and even if I have made a typing mistake the compiler’s gunna tell me about it up front and quick-smart.”

Another is chipping in with; “OK smarty-pants, but what if that SQL DBA changes either the table name or the column name in the SQL database?” To which I would reply; “I’m very glad you asked … because, yes, my app would ‘break’ (either way) – *but*, it would do so as soon as the SQL table was first `OPEN`ed and the ABC-generated error trapping would handle the problem ‘gracefully’. And then, and this is the good bit, I need only to replicate the name changes in the Clarion Dictionary and my Figure 13 code will continue to work, untouched. Whereas, every instance of Figure 12 will need to be found and fixed.”

I’m quite certain I have not convinced everyone out there that “raw SQL” is a sin, but I do hope this is food for thought.

Back to my self-set challenge: My premise was this; that I would (eventually, once I got the essentials sorted out) use a Template front end similar to the one provided by browse Totaling (see Figure 4) to specify the Totaling type required and the Field, from the View, on which I wanted the Totaling to operate. Therefore, from this interface I was assuming I'd end up with three template variables to work with; `%TotalingType`, `%FieldToTotal` and `%TotalTarget`.

Here's the resulting pseudocode, referencing these variables to achieve the required outcome;

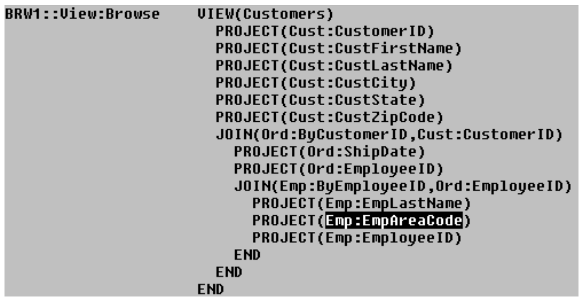
```
IF %TotalingType = 'COUNT' AND SELF.FileLoaded
  %TotalTarget = SELF.ListQueue.RECORDS() ! Row-Count = Records in the Queue
ELSE
  ! Hijack the WHERE-clause from the SQL-SELECT issued via the Browse's VIEW
  ! – resulting in SQLString = 'FROM ~~ WHERE ~~ ORDER BY 1'
  IF %TotalingType = 'COUNT'
    SQLString = 'SELECT COUNT(*) '& SQLString
  ELSE ! %TotalingType must be SUM or AVG
    ! Determine the ALIAS-reference to be applied to %FieldToTotal
    SQLString = 'SELECT '& %TotalingType &'('& ALIAS &'.'& %FieldToTotal &|
                ') '& SQLString
  END!IF
  SQLDummy{PROP:SQL} = SQLString ! Issue SQL-SELECT for COUNT, SUM or AVG
  NEXT(SQLDummy) ! Get the result of the SQL-SELECT into SQLDummy's buffer
  %TotalTarget = Result now in SQLDummy's buffer
END!IF
0{PROP:Text} = %TotalTarget ! Display the totaling result
```

Some clarifications of the pseudo code are in order here:

By my references to `SELF`, you can see that I'm assuming the code to be spawned by the (eventual) Template will be generated into one of the *`BrowseClassName`* procedures. In fact, the host method will be `BRWn.ResetQueue` … where I'm using `BRWn` to stand-in for *`BrowseClassName`* – 'cos I know that you'll know what I mean.

As covered earlier in this article, *if* the requirement is to *count* rows in the browse list *and* the browse list is file Loaded then that's simply achieved. Job done.

My test rig for this exercise is a `VIEW` that links three (SQL) tables. I aim to validate my solution by `SUM`ming the `Emp:EmpAreaCode` field … not because that makes any sense but because it's a numeric field so it can be `SUM`med, *and* because it's three deep in the `PROJECT/JOIN` structure; therefore, it will be a good test of the logic.



```
BRW1::View:Browse    VIEW(Customers)
                       PROJECT(Cust:CustomerID)
                       PROJECT(Cust:CustFirstName)
                       PROJECT(Cust:CustLastName)
                       PROJECT(Cust:CustCity)
                       PROJECT(Cust:CustState)
                       PROJECT(Cust:CustZipCode)
                       JOIN(Ord:ByCustomerID,Cust:CustomerID)
                         PROJECT(Ord:ShipDate)
                         PROJECT(Ord:EmployeeID)
                         JOIN(Emp:ByEmployeeID,Ord:EmployeeID)
                           PROJECT(Emp:EmpLastName)
                           PROJECT(Emp:EmpAreaCode)
                           PROJECT(Emp:EmployeeID)
                         END
                       END
                     END
```

Figure 14. The VIEW from which Emp:EmpAreaCode is to be SUMmed

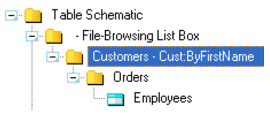
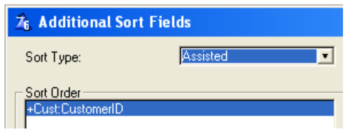


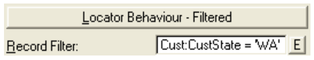
Figure 15. The Table Schematic that produces the VIEW shown in Figure 14



Sort Type: Assisted

Sort Order
+Cust:CustomerID

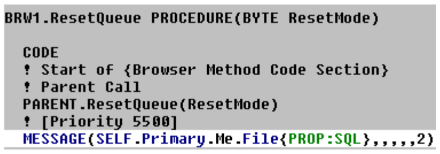
Figure 16. Additional ORDER BY to be applied to the VIEW shown in Figure 14



Record Filter: Cust:CustState = 'WA'

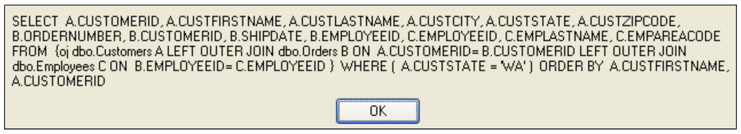
Figure 17. WHERE clause to be applied to the VIEW shown in Figure 14

This statement …



```
BRW1.ResetQueue PROCEDURE(BYTE ResetMode)

  CODE
  ! Start of {Browser Method Code Section}
  ! Parent Call
  PARENT.ResetQueue(ResetMode)
  ! [Priority 5500]
  MESSAGE(SELF.Primary.Me.File{PROP:SQL},,,,,2)
```

Results in this output … where Parameter=2 enables it to be copied to clipboard



```
SELECT  A.CUSTOMERID, A.CUSTFIRSTNAME, A.CUSTLASTNAME, A.CUSTCITY, A.CUSTSTATE, A.CUSTZIPCODE,
B.ORDERNUMBER, B.CUSTOMERID, B.SHIPDATE, B.EMPLOYEEID, C.EMPLOYEEID, C.EMPLASTNAME, C.EMPAREACODE
FROM  {oj dbo.Customers A LEFT OUTER JOIN dbo.Orders B ON  A.CUSTOMERID= B.CUSTOMERID LEFT OUTER JOIN
dbo.Employees C ON  B.EMPLOYEEID= C.EMPLOYEEID }  WHERE (  A.CUSTSTATE = 'WA' )  ORDER BY  A.CUSTFIRSTNAME,
A.CUSTOMERID
```

Figure 18. Example of SQL statement (issued by SQL Driver) on behalf of this VIEW

In the pseudo code I talk about "hijacking the `WHERE` clause" from the SQL `SELECT` statement that is issued (by the SQL Driver) on behalf of the `VIEW`. What I mean is that, in order for the `SUM` statement to be `SUM`ming(`%FieldToTotal`) from precisely the same selection of records as appears in the browse list, I need the `WHERE <condition-is-TRUE>` that's to be applied to the `SELECT SUM(%FieldToTotal)` to be exactly the same as the `WHERE` clause that was applied (by the SQL Driver) on behalf of the `VIEW`. The best way to achieve that is to borrow and adapt the very same `WHERE <condition-is-TRUE>` that actually was applied (by the SQL Driver) on behalf of the `VIEW`. **Tip:** Jump ahead to Figure 21, and compare it with Figure 18 to see what I mean.

Strictly speaking, the `ORDER BY 1` (referred to in the pseudo code above) is not really needed. What's *definitely* NOT wanted, though, is the original `ORDER BY` statement that was applied (by the SQL Driver) on behalf of the `VIEW`. This is due to a SQL syntax requirement/restriction that I won't go into detail about here. Suffice it to say that the original `ORDER BY` statement has to be removed, and I decided to replace it with the innocuous `ORDER BY 1` … although, I could just as well have removed it altogether.

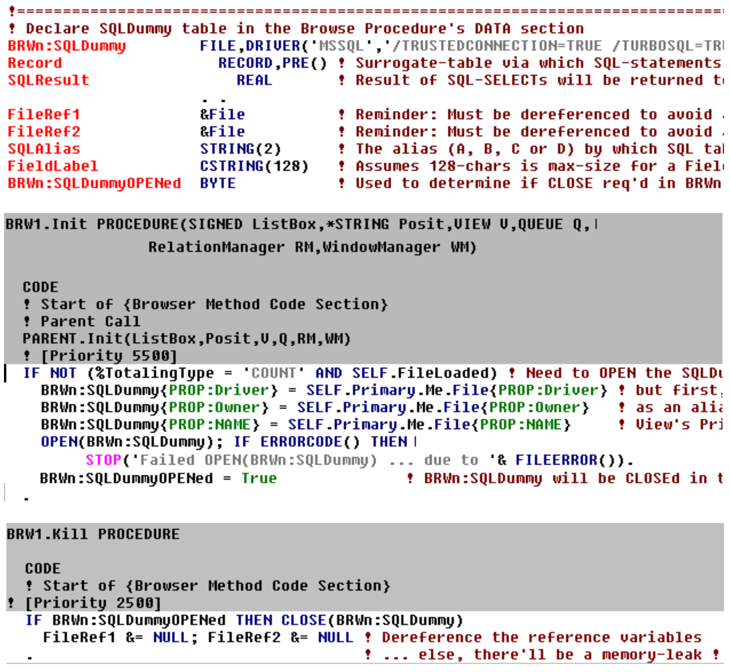
Later in the pseudo code I indicate the need to determine the `ALIAS` required to appropriately reference `%FieldToTotal`. You can find out more about this in the Clarion (6.3) Help Topic "SQL (use SQL code)", where the explanation, curiously, is more thorough than it is in the "PROP:Alias" topic.

The `ALIAS` is necessary because I am building up a SQL statement that must consist only of "raw SQL" – it cannot contain any Clarion-specific syntax. (At least, none that the SQL Driver cannot convert into "raw SQL" for me, but that's not relevant here – but if you are curious, see the Help Topic "Use of MATCH with PROP:Filter and SQL Databases".)

Accordingly, in the resulting SQL statement, each table field must have a unique reference. For example, there could be a `PhoneNumber` column in the `Customer` table and a same named `PhoneNumber` column in the `Employee` table. I cannot differentiate these as `Cust:PhoneNumber` and `Emp:PhoneNumber`, because that's Clarion-specific syntax and the SQL interpreter/compiler won't like it. The Clarion SQL Driver gets around this by defining an `ALIAS` for each table in the `VIEW` (`A` for the `VIEW`'s primary table, `B` for the next table `JOIN`ed into the `VIEW`, and so on). Therefore, using SQL's dot syntax, it's acceptable to refer to `A.PhoneNumber (= Customer.PhoneNumber)` and `B.PhoneNumber (= Employee.PhoneNumber)`. So, the task referred to here is the process of determining the `ALIAS` of the table (in the `VIEW`) to which `%FieldToTotal` belongs … `A.`, `B.` or `C.` etc.

Issuing the SQL `SELECT` statement for `COUNT`, `SUM` or `AVG` is achieved via `{PROP:SQL}`. You can see this `PROP`erty in action back in Figure 18, where it's used to regurgitate the last SQL statement issued by the SQL Driver.

The `{PROP:SQL}` must be issued via a SQL table *target*, which references any table in the SQL database, and not necessarily the table in question. I'm using a dummy table called `BRWn:SQLDummy` for this purpose.



```
!===========================================================================
! Declare SQLDummy table in the Browse Procedure's DATA section
BRWn:SQLDummy        FILE,DRIVER('MSSQL','/TRUSTEDCONNECTION=TRUE /TURBOSQL=TR
Record                 RECORD,PRE() ! Surrogate-table via which SQL-statements
SQLResult                REAL       ! Result of SQL-SELECTs will be returned t
                     . .
FileRef1             &File          ! Reminder: Must be dereferenced to avoid
FileRef2             &File          ! Reminder: Must be dereferenced to avoid
SQLAlias             STRING(2)      ! The alias (A, B, C or D) by which SQL ta
FieldLabel           CSTRING(128)   ! Assumes 128-chars is max-size for a Fiel
BRWn:SQLDummyOPENed  BYTE           ! Used to determine if CLOSE req'd in BRWn

BRW1.Init PROCEDURE(SIGNED ListBox,*STRING Posit,VIEW V,QUEUE Q,
                RelationManager RM,WindowManager WM)

  CODE
  ! Start of {Browser Method Code Section}
  ! Parent Call
  PARENT.Init(ListBox,Posit,V,Q,RM,WM)
  ! [Priority 5500]
  IF NOT (%TotalingType = 'COUNT' AND SELF.FileLoaded) ! Need to OPEN the SQLD
    BRWn:SQLDummy{PROP:Driver} = SELF.Primary.Me.File{PROP:Driver} ! but first
    BRWn:SQLDummy{PROP:Owner} = SELF.Primary.Me.File{PROP:Owner}   ! as an ali
    BRWn:SQLDummy{PROP:NAME} = SELF.Primary.Me.File{PROP:NAME}     ! View's Pri
    OPEN(BRWn:SQLDummy); IF ERRORCODE() THEN
         STOP('Failed OPEN(BRWn:SQLDummy) ... due to '& FILEERROR()).
    BRWn:SQLDummyOPENed = True                ! BRWn:SQLDummy will be CLOSEd in t
  .

BRW1.Kill PROCEDURE

  CODE
  ! Start of {Browser Method Code Section}
! [Priority 2500]
  IF BRWn:SQLDummyOPENed THEN CLOSE(BRWn:SQLDummy)
   FileRef1 &= NULL; FileRef2 &= NULL ! Dereference the reference variables
  .                                   ! ... else, there'll be a memory-leak !
```

Figure 19. Declaring & managing the SQL table via which SQL statement is to be issued

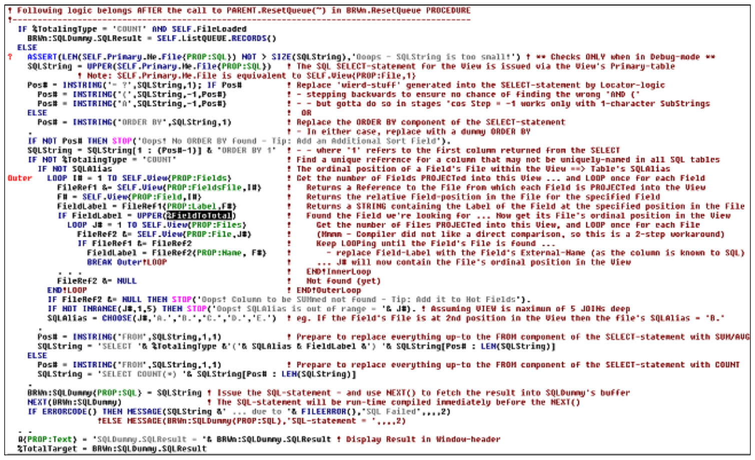
As the Help Topic for `PROP:SQL` explains, "If you issue an SQL statement that returns a result set (such as [via a] SQL SELECT statement), you use NEXT(file) to retrieve the result set one row at a time, into the file's record buffer." Normally, the SQL Driver insists that any result set must match the data type, `EXTERNAL` name and number of columns in the Target table. However, a handy SQL Driver enhancement that came out with Clarion Version 6.2 allows this (normally "good") requirement/restriction to be overridden. The driver attribute `/TURBOSQL` instructs the SQL Driver to;

"… simply define a buffer on the application side that will receive a PROP:SQL result set without the need to define a view with valid column name and data type." I have invoked this feature in my declaration of BRWn:SQLDummy table – See Figure 19.

It's the `NEXT(`*`Target`*`)` statement that initiates all the real work. The SQL Statement that was issued via the preceding *`Target`*`{PROP:SQL}` statement is compiled on-the-fly by the SQL Driver and passed on to the SQL environment identified by the *`Target`*. The Clarion run time then waits patiently for the result to be returned into the *`Target`*'s buffer – unless a problem is encountered, in which case `FILEERROR()` will contain an error message (not usually very helpful!) returned from the back end SQL server.

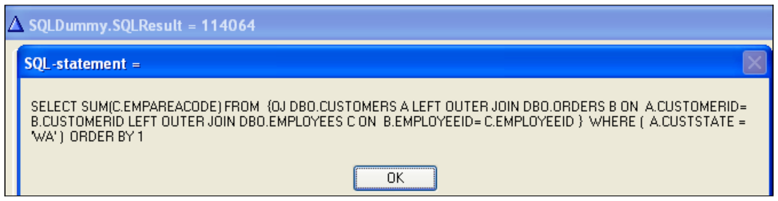
Finally, in my pseudo code, I'm using `0{PROP:Text}` to display the Totaling result … where zero specifies the *Target* as being the `Window` in which the browse resides.

Figure 20 fills in a lot of the gaps that I've glossed over in the pseudo code outline. For example, I found that use of a 'Filter Locator' results in a very strange SQL `SELECT` statement being generated – containing embedded, unquoted question marks. The code documented by reference to "Replacing 'weird stuff'" simply strips this out of the `SQLString`. There are some interesting references in the `LOOP` section too, where I'm obliged to apply some devious abstract references to find the `ALIAS` I need for `%FieldToTotal`.



```
! Following logic belongs AFTER the call to PARENT.ResetQueue(~) in BRWn.ResetQueue PROCEDURE
!-----------------------------------------------------------------------------------------------
  IF %TotalingType = 'COUNT' AND SELF.FileLoaded
    BRWn:SQLDummy.SQLResult = SELF.ListQUEUE.RECORDS()
  ELSE
?   ASSERT(LEN(SELF.Primary.Me.File{PROP:SQL}) NOT > SIZE(SQLString),'Ooops - SQLString is too small!') ! ** Checks ONLY when in Debug-mode **
    SQLString = UPPER(SELF.Primary.Me.File{PROP:SQL})    ! The SQL SELECT-statement for the View is issued via the View's Primary-table
              ! Note: SELF.Primary.Me.File is equivalent to SELF.View{PROP:File,1}
    Pos# = INSTRING('= ?',SQLString,1); IF Pos#          ! Replace 'wierd-stuff' generated into the SELECT-statement by Locator-logic
      Pos# = INSTRING('(',SQLString,-1,Pos#)             ! - stepping backwards to ensure no chance of finding the wrong 'AND ('
      Pos# = INSTRING('A',SQLString,-1,Pos#)             ! - - but gotta do so in stages 'cos Step = -1 works only with 1-character SubStrings
    ELSE                                                 !  OR
      Pos# = INSTRING('ORDER BY',SQLString,1)            ! Replace the ORDER BY component of the SELECT-statement
    .                                                    ! - In either case, replace with a dummy ORDER BY
    IF NOT Pos# THEN STOP('Oops! No ORDER BY found - Tip: Add an Additional Sort Field').
    SQLString = SQLString[1 : (Pos#-1)] & 'ORDER BY 1'   ! - - where '1' refers to the first column returned from the SELECT
    IF NOT %TotalingType = 'COUNT'                       ! Find a unique reference for a column that may not be uniquely-named in all SQL tables
      IF NOT SQLAlias                                    ! The ordinal position of a Field's File within the View ==> Table's SQLAlias
Outer   LOOP I# = 1 TO SELF.View{PROP:Fields}            ! Get the number of Fields PROJECTed into this View ... and LOOP once for each Field
          FileRef1 &= SELF.View{PROP:FieldsFile,I#}      !   Returns a Reference to the File from which each Field is PROJECTed into the View
          F# = SELF.View{PROP:Field,I#}                  !   Returns the relative Field-position in the File for the specified Field
          FieldLabel = FileRef1{PROP:Label,F#}           !   Returns a STRING containing the Label of the Field at the specified position in the File
          IF FieldLabel = UPPER(%FieldToTotal)           !   Found the Field we're looking for ... Now get its File's ordinal position in the View
            LOOP J# = 1 TO SELF.View{PROP:Files}         !     Get the number of Files PROJECTed into this View, and LOOP once for each File
              FileRef2 &= SELF.View{PROP:File,J#}        !     (Mmmm - Compiler did not like a direct comparison, so this is a 2-step workaround)
              IF FileRef1 &= FileRef2                    !     Keep LOOPing until the Field's File is found ...
                FieldLabel = FileRef2{PROP:Name, F#}     !       - replace Field-Label with the Field's External-Name (as the column is known to SQL)
                BREAK Outer!LOOP                         !     ... J# will now contain the File's ordinal position in the View
          . . .                                          !   END!InnerLoop
          FileRef2 &= NULL                               !   Not found (yet)
        END!LOOP                                         ! END!OuterLoop
        IF FileRef2 &= NULL THEN STOP('Oops! Column to be SUMmed not found - Tip: Add it to Hot Fields').
        IF NOT INRANGE(J#,1,5) THEN STOP('Oops! SQLAlias is out of range - '& J#). ! Assuming VIEW is maximum of 5 JOINs deep
        SQLAlias = CHOOSE(J#,'A.','B.','C.','D.','E.')  ! eg. If the Field's File is at 2nd position in the View then the file's SQLAlias = 'B.'
      .
      Pos# = INSTRING('FROM',SQLString,1,1)              ! Prepare to replace everything up-to the FROM component of the SELECT-statement with SUM/AVG
      SQLString = 'SELECT '& %TotalingType &'('& SQLAlias & FieldLabel &') '& SQLString[Pos# : LEN(SQLString)]
    ELSE
      Pos# = INSTRING('FROM',SQLString,1,1)              ! Prepare to replace everything up-to the FROM component of the SELECT-statement with COUNT
      SQLString = 'SELECT COUNT(*) '& SQLString[Pos# : LEN(SQLString)]
    .
    BRWn:SQLDummy{PROP:SQL} = SQLString ! Issue the SQL-statement - and use NEXT() to fetch the result into SQLDummy's buffer
    NEXT(BRWn:SQLDummy)                 ! The SQL-statement will be run-time compiled immediately before the NEXT()
    IF ERRORCODE() THEN MESSAGE(SQLString &' ... due to '& FILEERROR(),'SQL Failed',,,,2)
                  !ELSE MESSAGE(BRWn:SQLDummy{PROP:SQL},'SQL-statement = ',,,,2)
  . .
  0{PROP:Text} = 'SQLDummy.SQLResult = '& BRWn:SQLDummy.SQLResult ! Display Result in Window-header
  %TotalTarget = BRWn:SQLDummy.SQLResult
```

Figure 20. Semi pseudo code – as intended to be generated by a (yet to be written) Template (view full size image)



SQLDummy.SQLResult = 114064

SQL-statement =

SELECT SUM(C.EMPAREACODE) FROM {OJ DBO.CUSTOMERS A LEFT OUTER JOIN DBO.ORDERS B ON A.CUSTOMERID= B.CUSTOMERID LEFT OUTER JOIN DBO.EMPLOYEES C ON B.EMPLOYEEID= C.EMPLOYEEID } WHERE (A.CUSTSTATE = 'WA') ORDER BY 1

Figure 21. Example of generated SUM() where %FieldToTotal = Emp:EmpAreaCode

The result produced by the SQL statement in Figure 21 is instantaneous. And my browse list is not being double-filled.

"Whoa, hold on" I hear someone exclaim, indignantly; "You're committing the exact same 'sin' here that you proclaimed against earlier!" … And he'd be right. However, I consider it to be OK when the "raw SQL" is to be generated by a Template – because it won't be dependant upon me to get it right and I know I can test it thoroughly just once and it will be correct forever. And, even if 'forever' is sooner than I expect, I can fix any bug in the Template code knowing it will be automagically corrected everywhere, simply by regenerating the code created by the template. I can live with that, conscience free.

However, I haven't actually got to this stage just yet, (where my work can be replicated by a Template), and I'm now quite wearied by my long journey … to be honest, I'm flagging.

Instead, I'll offer this skeleton to the Clarion community in the hope that someone will be interested in applying their template-building skills to it. If so, we'll *all* be able to have our cake and eat it too.

John Morter is Asia Pacific IT Manager for a brand-name multi-national and he's supposed to leave all the fun technical stuff for others to do. As a result, his Clarion work is developed under the nom-de-keyboard Flat Chat Solutions, where "flat chat" is an Australian expression meaning doing something at top speed / high velocity. John's inspirational hero is David Bayliss, who was the principal architect of and evangelist for the ABC Libraries.

Article comments

BACK TO TOP

SV Announces 2011 Clarion DevCon

Posted April 29 2011

It's official: there will be a SoftVelocity Clarion DevCon this year. The event will be held in Denver Colorado, November 11-13, 2011.

The conference location is the same as it was last year for the ClarionLive DevCon. But it isn't just a regional, independent conference: it's the sanctioned Clarion conference of the year, co-organized by SoftVelocity and ClarionLive (Arnold Young, John Hickey). Bob Zaunere will be present, along with selected SoftVelocity staff.

Further information will be posted at the ClarionLive and SoftVelocity web sites in the coming months, and of course Clarion Magazine will keep you updated.

The conference venue is the Inverness Hotel and Conference Center, which received rave reviews from attendees at the '10 ClarionLive event.

Article comments

BACK TO TOP

A String Class

By Steven Parker

Posted April 29 2011

It has been a bit over eight years since I wrote my first class (“A Class for Tagging”). In that article, I observed that I broke down and wrote a class out of sheer laziness. “I *use* OOP ... but I do not *do* OOP.”

And I haven’t written a class since. Until now. And, this time, necessity was the prime mover.

My Sad Story

How I got forced to create my second class turns on an application that the owner/end-user handed over to me for maintenance and updating.

One of the things the app does is categorize animals by a code. This animal code field is a three digit number, loosely based on codes supplied by a national certifying association.

Nothing remarkable about this field except that when I first looked at the field’s usage, I noticed it was not a look up from the national association’s code file; now, *that* was interesting. Nothing (else) remarkable until the national organization notified my customer that they were going to start using four digit codes.

Well, there was *one* remarkable thing. My customer did not use the national organization’s codes directly. My customer’s app mapped the national codes to their own proprietary codes (their codes paired with the national association’s codes in the animal code master list but, in the animal file, only my customer’s codes are used – “but that’s another story”). This allows my customer to use AnimalCodes as he wants; specifically, even hundreds (100, 200, 300, etc.) are used as “category” names within his proprietary number scheme. This observation, which I had not yet made, was to play a major role in what transpired.

I figured this was easy money. I could do it while occupied with TV. Generate the apps in the product suite, there are 11 or 12. Then, I would use Clarion Source Search to locate all the procedures where the variable is used. Finally, go into all the window and report structures and change the “@N3” to “@N4.” (And change the entry field width if really necessary.) No strain and everybody’s happy.

Well, I might have to adjust positions on some other controls, if the new length of the animal code crowded nearby controls. But, again, not rocket science.

Then, something really interesting happened. The first window I looked at showed the code field being formatted @S3. That’s odd. Well, it was an old app, after all….

So I adjusted it to @N4 and compiled. It ran just fine. But I could not enter “1234” on the update form window. It truncated to “123.” No matter what I did, the end result was three digits long.

A bit of investigating (and a lot of … unpublishable language) later, I saw that, in addition to some “interesting” validation code, the code field had been declared, in the dictionary, as a STRING(3). It had *never* occurred to me a code field might be declared as anything other than a LONG and, therefore, I didn’t check the dictionary. Go ahead, take a moment, commiserate with my “unpublishable language.”

Why Use Strings for Codes

Why would anyone use a String for a code field? The *only* condition under which I would consider a STRING for a code is if there were codes that contained, in fact *had* to contain, alpha characters. But I was assured that there were no alphameric codes in the system, never had been. Neither were there alphameric codes from the national organization, never had been.

So, *why* a string?

This is when I discovered that the even hundreds animal codes were being used to create an *ad hoc* category. All animals from 101 to 199 were in one category; 201 to 299 in another and so on.

For example:

```
100      Bovines
101      Herford
103      Longhorn
200      Equines
217      Quarter Horses
223      Pintos
Etc.
```

This allowed the previous developer to use this information to do dynamic group breaks on the “category” and print section headers in reports by comparing the current record’s animal code to the previous record’s animal code. The break was computed as follows:

```
If SAV:Code <> AnimalCode[1]
  Print(RPT:CategoryHeader)
  SAV:Code = AnimalCode[1]
End
```

Now, using `STRING(3)` makes sense. It is not what I would have done but at least I understand the thinking behind using a string for the code.

What I Would Have Done

What would I have done? Because I would have initially defined the `AnimalCode` field as a `LONG`, I could not have compared first digits. Clarion offers no built in way to get the first character of a number.

Yes, I could Cast my `LONG AnimalCode` to a `STRING`. Clarion’s automatic type conversion makes this easy. And, as long as animal codes are all the same length, the existing code is entirely workable with only minor modification.

What I have in mind is a `GetFirstCharacter` procedure that would work as follows:

```
GetFirstChar      Procedure(Long pCode)
LOC:String  String(256)
RetVal      String(1)

        CODE
 LOC:String = pCode
 RetVal = LOC:String[1]
 Clear(LOC:String)
 Return RetVal
```

So, the code could become something like:

```
If SAV:Code <> GetFirstChar(AnimalCode)
  Print(RPT:CategoryHeader)
  SAV:Code = GetFirstChar(AnimalCode)
End
```

Clarion converts my `LONG` to a `STRING` at the first line of the code. Then I can use standard string slicing to get and return the first character.

Once codes go to four characters, however, testing the first character completely breaks down. At four digits, 172 and 1239 have the same first digit and this is wrong.

172 and 1239 would end up printing the same category header (because that first character is used to reassemble the “even hundred” for a look up into the master code table). Additionally, expanding the existing STRING field to (4), the report order gets seriously dicey as strings and numbers do not sort the same; 10xx, as a string, sorts before 2xx.

Alternate Designs – How I Would Have Done It

A more general design would have declared `AnimalCode` as a `LONG`, as God and Mssrs. Date and Codd decreed. Then, I could extract and save the “hundreds” and use that for comparison:

```
If SAV:Code <> (AnimalCode / 100) * 100
  Print(RPT:CategoryHeader)
  SAV:Code = (AnimalCode / 100) * 100
End
```

(In this case, both `SAV:Code` and `AnimalCode` would be defined as `LONG`s. In the existing code, `SAV:Code` is `STRING(1)` and `AnimalCode` is a `STRING(3)`, remember.)

Assuming data types which do not support decimals (looking forward, a bad assumption), this code constructs the “hundreds” very nicely. Suppose `AnimalCode` is 472:

```
472 / 100 =  4.72
4.72 * 100  … whoops!
```

The automatic data conversion (and, yes, this is documented, is to a `REAL`).

Either I have to do an intermediate assignment of the division operation to a non-decimal type or I need to `INT` the division:

```
If SAV:Code <> INT(AnimalCode / 100) * 100
  Print(RPT:CategoryHeader)
  SAV:Code = INT(AnimalCode / 100) * 100
End
```

Another alternative, a variant on the code above, is to save the `AnimalCode`, as is, and extract the hundreds from each variable:

```
If INT(SAV:Code / 100) * 100 <> INT(AnimalCode / 100) * 100
  Print(RPT:CategoryHeader)
  SAV:Code = AnimalCode
End
```

Which Gives Me an Idea ...

I just hate refactoring my own code. "Refactor" $=^{def}$ "rethink, redesign, rewrite." I don't like the "rewrite" part.

How much more so when working with someone else's code! Especially before I've entirely mastered what my predecessor did and how s/he did it. Whenever I get clever with code I don't know intimately (and sometimes even when I do), things break. (The Law of Unintended Consequences truly is inexorable.)

But "how I would have done it" gives me an idea.

I need to redefine `SAV:Code`, that's obvious. Making `SAV:Code` a `LONG` or making it a longer `STRING` is a not a major change (finding all the occurrences, yes, changing, no). Though, a `LONG`, I think, will be more flexible in the end. So, that decision is made.

With `SAV:Code` as a `LONG`, it is able to handle three and four digit animal codes (whether I decide to store/compare the whole code or just the hundreds portion of it). And I have already sketched out two different ways of comparing AnimalCodes.

I just need to decide between the two techniques "I would have done."

And, then, I need to implement it … in several hundred places.

But wait! There's more …

Of course, I found other code-like fields declared as strings. I found first character comparisons involving those fields. I found first three character comparisons. I found … many additional challenges.

Because the additional code-types I found are 100% proprietary, I can leave these alone for the time being. But I am more comfortable preparing now for what I know to be inevitable. Inevitably, these codes will have to go numeric also. But not just now. (Mark Donahue's variant on Murphy's Law: If it ain't broke, it soon will be.)

Enter the dragon

All I have to do is change `SAV:Code` to a `LONG` and, then, find all instance of

```
If SAV:Code <>  AnimalCode[1]
```

(and similar code), replacing it with

```
If SAV:Code <>  INT(AnimalCode / 100) * 100
```

In this case, I also need to load the hundreds value into `SAV:Code`, like:

```
SAV:Code = INT(AnimalCode / 100) * 100
```

or with

```
If INT(SAV:Code /  100) * 100 <> INT(AnimalCode / 100) * 100
```

In this case, I simply need to change the current:

```
SAV:Code = AnimalCode[1]
```

to

```
SAV:Code = AnimalCode
```

several hundred times.

Suddenly, the second method, computing hundreds on both variables and comparing the results, looks a bit more appealing.

Choices

My first alternative is to "just" (1) find and replace the declarations of `SAV:Code`, (2) find/replace the assignment to `SAV:Code` and (3) find and retype all the instances of `SAV:Code <> AnimalCode[1]`.

Sounds like a lot of typing. Nothing can be done for (1) and (2); those tasks have to be done. But there has to be a better way than typing

```
If (SAV:Code / 100)  * 100 <> (AnimalCode / 100) * 100
```

hundreds of times. This, of course, means many opportunities for typographical errors. (Cut and paste would work if the save variable were always "`SAV:Code`." It isn't; a second set of typo-tunities.)

A purpose-built function seems the way to go. And that means a class (because a DLL seems like overkill; besides, including and instantiating a class really is easier than including an External DLL with multiple exported procedures).

The Function

I can think of two ways of doing this.

Method 1: I could create a function that returns the “hundreds” of a number. I would call it for each number and compare the results:

```
If MyClass.Hundreds(pCode1)  <> MyClass.Hundreds(pCode2)
```

Method 2: I could create a function that takes both numbers, converts them to their “hundreds,” compares them and tells me if they are the same:

```
If  ~MyClass.CompareHundreds(pCode1, pCode2)
```

(If you’re thinking to yourself, “but this won’t work for 1-99,” quite so. Fortunately, `AnimalCodes` less than 100 are not allowed.)

Thinking about it, `MyClass.CompareHundreds` is easy to write. It should look like (pseudo-code):

```
If Self.Hundreds(pCode1) = Self.Hundreds(pCode2)
  Return True
Else
  Return False
End
```

The `Hundreds` function isn’t difficult either, given that I’ve already written it above:

```
Return (INT(pCode/100)  * 100 )
```

But First ...

There is a decision to be made. That decision is “What data types is the function to receive?”

I know that I will be converting the `AnimalCode` file fields from `STRING` to `LONG` in the DCT (another real joy as the files are Btrieve). But I also know that there may be (actually, are) fields that will figure in various computations that are `STRING`s (and will, at least for the foreseeable future, remain `STRING`s).

So, do I create a function that take two `LONG`s? Then overload that function with an additional function that take two `STRING`s? Then, there are cases where I might have one of each (and in different orders). Total? At least four functions.

I don’t think so. Clarion wouldn’t make me do that!

Indeed, Clarion doesn’t. Clarion has a special data type. The “I’ll decide what I am at run time” data type (a quantum data type?). Formally known as a “Parameter of Unspecified Data Type,” an “ANY,” it is prototyped with a question mark.

As the on line help says:

> Within the procedure, the parameter acts as a data object of the type of the variable passed in at runtime. This means the data type of the parameter is fixed during the execution of the `PROCEDURE`.

Inside my code, I will be assigning incoming parameters to local variables or I will be doing arithmetic operations on them (as in the Hundreds function, above). Clarion’s automatic type conversion will ensure that, regardless of the nature of the input, the computations and output will be just what I expect (assuming I read the documentation on automatic type conversions; if I don’t, they computations will at least be predictable).

Just what the good doctor needs to keep his class under control.

Building the Class

First, the INC file, the list of data used in the class and the prototypes of the procedures that compose the class.

I start with the standard “header:”

```
OMIT('_EndOfInclude_',_SClass_)
_SClass_ EQUATE(1)
```

With “`_EndOfInclude_`” at the end of the file, the class can be included only once in an app.

Next, I declare the class, which I’ve decided to call “`StringCompare`:”

```
!StringCompare    CLASS,TYPE,MODULE('StringClass.clw'),LINK('StringClass.clw',_ABCLinkMode_),DLL(_ABCDl
StringCompare    CLASS,TYPE,MODULE('StringClass.clw'),LINK('StringClass.clw')
```

Note that initially I used `_ABCLinkMode_` and `_ABCDllMode_`. I rethought that.

Using these attributes, I must include the class in the root DLL of any multi-DLL app (as I quite soon discovered). I don’t need to pass the class from .APP to .APP. I only need the class’ functions in the .APPs where I need to do the comparisons mentioned. In other words, I don't need the capabilities that these attributes offer.

So, I removed them. Without them, I only need to include the class in those places where I actually intend to use it (two or three of the apps in the suite).

Next, I automatically include:

```
Construct         Procedure
Destruct          Procedure
```

in the procedure list because I just don't know whether I'll need a constructor or not. Easier just to put these two methods there from the beginning.

Now, I prototype the two functions I know I'll be using immediately:

```
Huns           Procedure(? pCode),Long
CompareHuns    Procedure(? pLeftField, ? pRightField),Byte
```

("Huns," granted not the most … uh, intuitive of method names. But, unfortunately, this customer is still in 6.1 and 6.1 does not have the code completion feature of 7.x which would allow me to use longer, more descriptive Labels without fear of excessive typing.)

I also know I need procedures – for other purposes -- to get the first character of a STRING or of a numeric and some other functions. So, the final prototype list looks like:

```
CompareFirstChar  Procedure(? pStringField, ? pLongField),Byte
Huns              Procedure(? pCode),Long
GetFirstChar      Procedure(? pCode),String
ReplaceText       Procedure(STRING pOldStr, STRING pNewStr, STRING pText),String
Num2String        Procedure(? pNumber),String
GetFirstNonBlank  Procedure(? pStringIn),String
CompareHuns       Procedure(? pLeftField, ? pRightField),Byte
CLen              Procedure(String pString),Long
              End

_EndOfInclude_
```

N.B.: The `ReplaceText` function is Peter Hermansen's. I found it in my knowledge base and figured "what the heck!"

The CLW file

This file contains the functioning code. It starts pretty much the way all class CLWs start (given that I copied the first few lines from a shipping class files, this isn't surprising):

```
   MEMBER

    Include('StringClass.inc'),ONCE

   Map
   End
```

The use of the Map statement ensures that standard Clarion functions (from BUILTINS.CLW) are linked in.

The `Construct` and `Destruct` methods are easy because I don't currently have anything I need to do in them:

```
StringCompare.Construct          Procedure

   Code

StringCompare.Destruct           Procedure

   Code
```

Now, I must code the function to compute and return the "hundreds" of a passed datum. This looks just like described earlier:

```
StringCompare.Huns            Procedure(? pCode) !,Long

         CODE
 Return ( INT(pCode/100) * 100  )
```

Likewise, the function to compare the "hundreds" component of two data is exactly as described earlier:

```
StringCompare.CompareHuns    Procedure(? pLeftField, ? pRightField) !,Byte
RetVal Byte
```

```
       CODE
  If Self.Huns(pLeftField) = Self.Huns(pRightField)
    RetVal = True !Return True
  Else
    RetVal = False !Return False
  End
  RETURN RetVal
```

If you're interested in the remaining functions listed in the INC, see the source code, downloadable at the end of this article, in StringClass.CLW.

The only hard part is in `StringCompare.Huns`. The difficulty is remembering that this procedure is "data agnostic."

The data type of the pass parameter is determined at run time. So, the code could be operating on virtually any data type, a string, cstring, decimal, real, etc.

Because I only want an integer (2367 coming in returns 2300, etc.), I have to code to the possibility of decimals. Thus, INT.

The Demo App

Tester.APP demonstrates the procedures in the class.

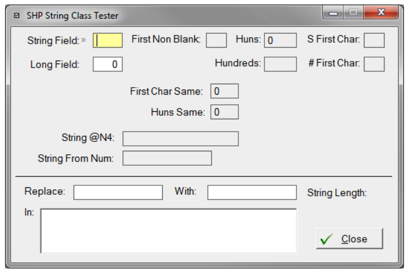


Figure 1. The demo app

The first entry field is a `STRING`, the second is a `LONG`. When you make entries in both fields, the class will compute and display:

- The first character of each field
- The hundreds component of each field
- The first non-blank character of the `STRING` field (I got carried away …)
- Whether the first character is the same
- Whether the hundreds component is identical

There are also a couple of other items displayed (I needed to see how certain entries would format). The important thing is that I made a decision in the `CompareFirstCharacter` method. I decided that "123" does equal 123 but that " 123" (leading space) does not. To compare based on the first non-blank character, restore the line in the CLW that includes "! ignoring leading spaces."

I suppose I could rewrite the `CompareFirstCharacter` method to take an additional flag, "`IgnoreLeadingBlanks`" and, thus, have a completely generic function. But my ambitions do have limits.

Also note, I made another decision in the `CompareFirstCharacter` method. I decided I didn't need to check any further if the non-blank lengths of the two data were not equal. If the lengths didn't match, I was obviously comparing a three digit `AnimalCode` to a four digit code. As discussed above, such a comparison cannot return a true value, so no need to check further.

If you don't have C7 yet, I have included a compiled EXE and all the necessary DLLs to run it. I have also included TXAs (APVs) so you can find the code.

The actual calls, all in one embed, are:

```
 If StringField and LongField
   N4Format = '|' & Format(StringField,@n4) & '|'
   Hundreds = sCompare.Huns(LongField)
   Huns = sCompare.Huns(StringField)
   FirstCharSame = sCompare.CompareFirstChar(StringField,LongField)
   HunsSame = sCompare.CompareHuns(StringField,LongField)
   FirstNonBlank = sCompare.GetFirstNonBlank(StringField)
   StringFromNum = sCompare.Num2String(LongField)
   FirstChar = sCompare.GetFirstChar(LongField)
   sFirstChar = sCompare.GetFirstChar(StringField)
   ThisWindow.Reset(1)
 End
```

Summary

Has class writing gotten any easier in eight years? Nope. Not hardly. I made every mistake on this one that I did on the first one. At least this time, I had "eight year ago 'me'" to consult (knowing full well he was laughing at me).

The big lesson is that I saw the need for new code and immediately thought "class!" (a real breakthrough for me). And, this time, I understood why I made that choice:

- the code would be used in multiple places within a single .APP
- the code would be used in multiple .APPs

and

- the code is of the type that I would have included in function library (DLL) in the past.

Download the source

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

Article comments

by Bob Roos on May 1 2011 (comment link)

Looking at ReplaceText, why doesn't it blow up if you replace 1 character with 10 characters? Does Clarion keep expanding pText?

Also what happens if the new string contains the old? Self.ReplaceText('100','10100',MyString)

by Steven Parker on May 1 2011 (comment link)

Bob, as I said in the article, I ... requisitioned this code from someone else.

Your questions are good ones. Sounds like a bit of testing on your part and a follow-on article from you .

BACK TO TOP

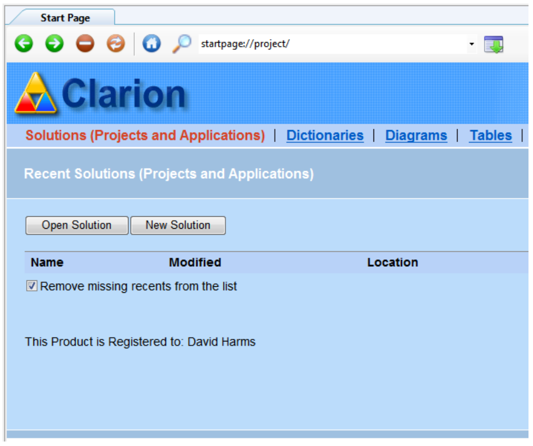
Clarion 8: First Look

By Dave Harms

Posted April 30 2011

The first Clarion 8 release, build 8255, is now available from SoftVelocity. Emails have gone out to CSP participants.

Visually C8 has a more contemporary look, although as with any visual change I'm sure there will be a certain amount of disagreement over the aesthetics. Figure 1 shows a portion of the new start page.

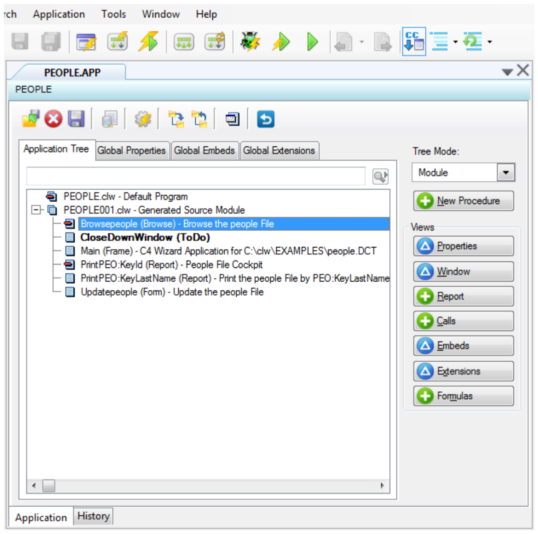

Start Page
startpage://project/
Clarion
Solutions (Projects and Applications) | Dictionaries | Diagrams | Tables |
Recent Solutions (Projects and Applications)
Open Solution
New Solution
Name
Modified
Location

☑ Remove missing recents from the list

This Product is Registered to: David Harms

Figure 1. The C8 start page

The new icons are immediately obvious, as in Figure 2. Some toolbar icons have been removed, and the icons themselves are somewhat larger.



PEOPLE.clw - Default Program
PEOPLE001.clw - Generated Source Module
Browsepeople (Browse) - Browse the people File
CloseDownWindow (ToDo)
Main (Frame) - C4 Wizard Application for C:\clw\EXAMPLES\people.DCT
PrintPEO:KeyId (Report) - People File Cockpit
PrintPEO:KeyLastName (Report) - Print the people File by PEO:KeyLastName
Updatepeople (Form) - Update the people File

Figure 2. The AppGen and part of the new toolbar

Installation

Installation went smoothly and was just like the C7 install process, except that the templates were not automatically registered, so I had to do that step manually.

I didn't notice a new features help topic on installation (although I did eventually find it), so rather than exploring the IDE and happening upon changes by chance, I decided for the purposes of this review to walk through the extensive change log for this release. All the change log items are in italics; my comments are in regular text. You can read my overall impressions at the end of this article.

New features

ABC Utility class DynaStringClass to declare dynamic strings with auto-dispose. DynaStringClass is a wrapper for a dynamically created CString, not to be confused with the mysterious IDynStr interface that has been in Clarion for some time now.

Ability to Run the Selected Application's Project Target from the Application Pad. A new button on the App Pad, beside the Build and Run button.

Activate use of hot keys for menu items when in the application editor. I haven't tested this much but it does seem like hotkeys work better now when you're in the AppPad. Menus still don't clear, however, as when you drop down a menu and then click on in the App editor window.

Added gradient support to TOOLBARs.

Added accelerator keys throughout the dictionary editor. More shortcut keys are always welcome.

Added a new redirection symbol %THISDIR% which is set to the directory of the redirection file. This is useful when you have nested redirection files and want to find a file based on the location of the redirection file. I can see this being a very handy feature. Not only useful for nested redirection files, but also useful when you have a work copy of a directory and you want some paths to be relative to the copy in which you're working.

Allow Data Pad items to be edited as text (in TXA format). This looks pretty nice. There's a button at the top of the data pad that launches an editor containing the TXA version of the chosen local/module/global data.

All Dictionary lists now have a Locator control. There are, in fact, locators everywhere in the dictionary. Users of very large dictionaries should find this quite helpful.

Application Options are now saved right after the dialog is closed not when the IDE is closed. This should mean an end to having to redo app options if the IDE crashes after I've made a change.

BOX and PANEL control now support gradient color using PROP:GradientFromColor, PROP:GradientToColor, PROP:GradientType. These appear to be runtime only - I don't see any support in the window designer.

Built-in support of HTML Help, with support to integrate additional help systems based on other formats. Good news!

C8 now inherits all Clarion versions set up in Clarion 7. As you'd expect; C8 is an evolution of C7 and the upgrade process should be much simpler than from C6 to C7.

Conversion of data files with large number of records from within the IDE is much faster.

Applications Pad displays the Target type (EXE or DLL) next to the App name. I hadn't noticed that right away. It's a nice touch.

Editing dependencies from the Application Pad will pre select the Selected Application Project in the Dependency Editor.

Evaluation of AND and OR expressions in EVALUATE (or FILTER) statements are now short-circuited (only the left hand side is evaluated if possible). Presumably this could help performance in some slow-running situations.

Three-state Checkbox (Checked,Unchecked,Indeterminate). From the help:

> The CHECK control can offer either two states (ON or OFF) or three states (ON, OFF and INDETERMINATE).
>
> By default, when the CHECK is unchecked (off) the USE variable receives a value of zero (0); when the CHECK is checked (on), the USE variable receives a value of one (1), and when the CHECK is in indeterminate state the USE variable receives a value of two (2) . The VALUE attribute can be used to change the default ON/OFF values, and set the USE variable to other values. You can also use the runtime properties PROP:TrueValue and PROP:FalseValue.
>
> If the STATE3 attribute is enabled then you have a three-state checkbox. You can specify the value the USE variable receives in the Designer or at runtime with PROP:State3Value.

Global Setting to force the save/restore of the Solution tree state after generation if the Solution pad is visible. I'm thinking I don't rely on the state of the Solution Tree as much as some people do...

IDE: improved icons, new Tab style, remove unnecessary toolbar buttons. The toolbar has been simplified and the icons are now larger. The bottom tabs are slightly thinner than in C7 on my machine, but that's because my icons are missing. A number of other users have reported a similar problem with missing tab icons, prompting some to suggest the icons should be optional. Even with the missing icons the net result is still a slight loss of working area.

If you select a .dctx file when doing File/Save As for a dictionary, then the dictionary will be saved as an XML representation of the dictionary. I'm not a big fan of using file extensions and file extensions only as a way to determine the saved file format - I'd prefer to see a Save As DCTX. But if it works, it works.

If a WindowPreviewer.inc file can be found via redirection, this file will be included into the generated program used to preview a window. See below for more on the WindowPreviewer.inc file.

If you add a trailing | to a path in a redirection file, this will inform the redirection system to not search any further when trying to open a file that matches the pattern at the start of the line. I don't know how many people will use this, but more power can't be a bad thing.

Improve speed on generation of multiple apps by only refreshing the Solution Tree at completion.

In the Dependency Editor the Sorted List of Project is synchronized with the Selected Project in the DropList of Projects.

New button in the Applications Pad to change the Generation order calling the Dependency Editor. This one is a bit unclear to me. I see three new buttons on the AppPad: Run the selected app's Project, Open the containing folder, and Refresh.

New PROP:NoThemed support for a SHEET control to allow transparent SHEETs when the application is using a manifest.

New Application Manifest Option to set PROP:NoTheme = True to all SHEET controls in the application. I'll leave the manifest comments to those who feel this is their destiny.

New WindowPreviewer.inc file creating template for easily creating WindowPreviewer.inc files. Now, this is interesting. You get at this template by choosing File | New | File and choosing the window previewer include file template. You get this:

```
  SECTION('Includes')
!Put variable and equate declarations here

  SECTION('AfterOpen')
!Put code that needs to be executed after the window is opened, but before entering the ACCEPT
  SECTION('InAccept')
!Put code here that is executed at the top of the ACCEPT loop
!if you set windowPreviewerCanExit__ to FALSE, the generated code will not
!check to see if a button was pressed and exit the preview
!Note that windowPreviewerCanExit__ is reset to TRUE every iteration of the ACCEPT loop
```

Among other things, you can use this to apply runtime gradients to your previewed windows. I have a window with a toolbar, and I created a WindowPreviewer.inc with this code:

```
  SECTION('AfterOpen')
    ?toolbar1{PROP:GradientFromColor} = 0BBBB00h
    ?toolbar1{PROP:GradientToColor} = 0FFFF00h
    ?toolbar1{PROP:GradientType} =  GradientTypes:VerticalCylinder
```

Figure 3 shows the preview window.

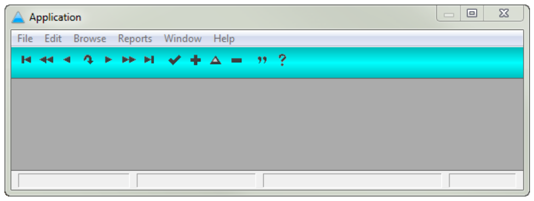


Figure 3. The window previewer with a runtime toolbar gradient

Unfortunately, that now assumes that I always have a ?toolbar1 field equate, so either more dynamic code is needed (detect a toolbar and apply the gradient if present) or perhaps there's a way to create window-specific .inc files.

New button in the Applications Pad to change the Generation order calling the Dependency Editor (EE only). A repeat of the above item, except this is EE only. I'm still unclear on the concept. And I have EE.

New algorithm provides much better look for disabled Icons and Images.

New Application Manifest Option to prevent using the Themed controls but still generate the .manifest file.

New SYSTEM property to get or set the Help Engine. Necessary, since HTML help is now an option.

New "Clear All Errors" button in the Errors pad. I'm glad to see this. I've often seen leftover errors.

Open File Dialog now show the extension sorted alphabetically.

Open File Toolbar button now has a drop option to show the QuickFile open extension.

Open the selected Application's Containing Folder from the Application Pad. I'm a fan of all of these different ways to open the relevant folder. In C8 and in C7 you can right-click on an open app's tab and get a context menu with this option.

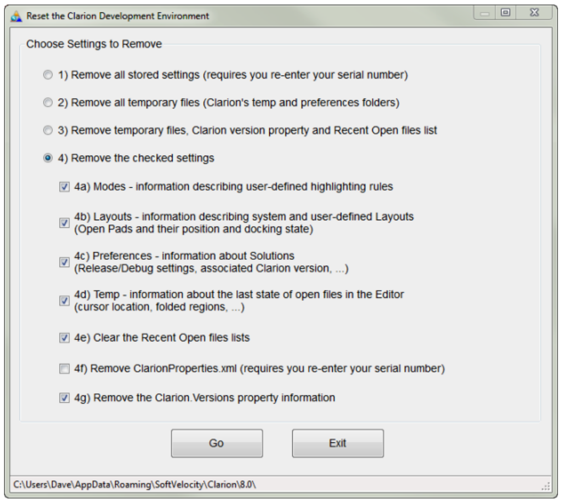
Pressing the alt keys to match the main menu items will now activate the main menus when you are in AppGen. This does work much more reliably in C8.

Pressing Ctrl-Enter when in a list in the Data pad or Dictionary editor will search to the next entry matching the contents of the locator. This is more or less a standard advance-the-locator feature in the Clarion IDE (or has been so traditionally), but there are now many more locators and so many more places to use this.

READONLY property is added to CheckBox and Option controls in the Designer (corresponds to the runtime property PROP:READONLY).

READONLY support for OPTION control.

ResetIDE.EXE tool to allow cleanup of the IDE settings and temp files. You need to look for this in the Bin directory - no link is provided. The tool provides a number of cleanup options, as shown.



Reset the Clarion Development Environment

Choose Settings to Remove

○ 1) Remove all stored settings (requires you re-enter your serial number)

○ 2) Remove all temporary files (Clarion's temp and preferences folders)

○ 3) Remove temporary files, Clarion version property and Recent Open files list

● 4) Remove the checked settings

- ☑ 4a) Modes - information describing user-defined highlighting rules
- ☑ 4b) Layouts - information describing system and user-defined Layouts (Open Pads and their position and docking state)
- ☑ 4c) Preferences - information about Solutions (Release/Debug settings, associated Clarion version, ...)
- ☑ 4d) Temp - information about the last state of open files in the Editor (cursor location, folded regions, ...)
- ☑ 4e) Clear the Recent Open files lists
- ☐ 4f) Remove ClarionProperties.xml (requires you re-enter your serial number)
- ☑ 4g) Remove the Clarion.Versions property information

C:\Users\Dave\AppData\Roaming\SoftVelocity\Clarion\8.0\

Figure 4. The ResetIDE tool

SLIDER Control support in the Designer. This is pretty nifty. In Figure 5 I've added a slider to a form, for no particular reason. Oddly the slider tab doesn't show in the default configuration.

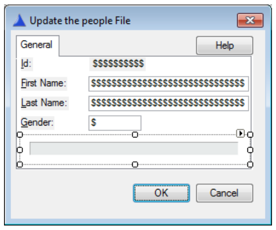


Figure 5. Adding a slider control.

Figure 6 shows the slider in the previewer.

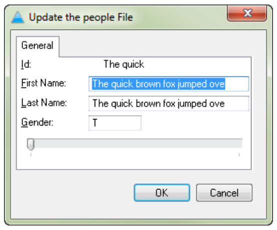


Figure 6. The slider in the previewer window.

Several different slider styles are available.

STATE3 supported in designer for CHECK controls.

Save the last state of nodes in the Embed tree. This has been on a lot of wish lists. In C7, every time you exit the embed list the IDE forgets where you were, and if you want to go back to that embed you have to hunt for it. In C8 the IDE remembers the last embed you were in *and* it remembers the state of the embed list, which nodes were expanded/collapsed. This holds true for as long as you have the app open.

The Audit and Comments tab now uses a splitter to allow the Audit details to be contracted.

The Reference check box will now preserve its original state if you scroll through the list of data types in the data type combo-box. Probably more of a bug fix than a feature. No doubt that will save a few headaches.

The Clarion linker now tells you in which two files the symbol exist in when getting a duplicate symbol error. Another time saver.

The Binary File Import process now prompts before overwriting the binary file with the text file. This is part of the Subversion version control integration system.

The position of the splitter between the Audit and Comments is remembered between windows and sessions.

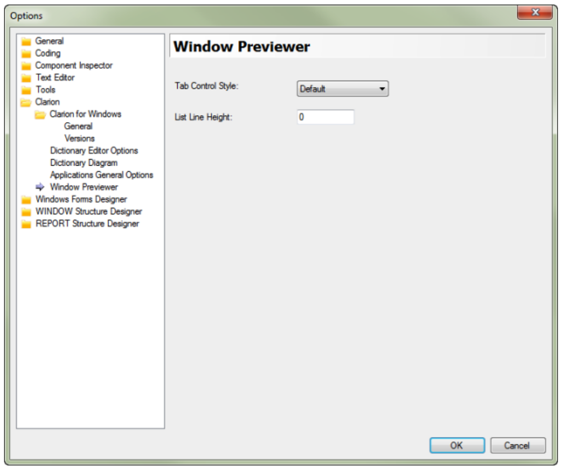
Tools->Options are now saved right after the dialog is Accepted not when the IDE is closed. Also the case for Application Options, as noted above.

Typing into a list in either the Data pad or the Dictionary editor fills the locator. Meaning, you don't have to have the focus on the locator before you start typing.

When the WINDOW/APPLICATION Structure have a width or height bigger than the screen resolution it will inform the user and keep the value till the value is changed by manually changing the designer or the property value in the property grid. I believe this was a fix to one of the later C7 builds. Before this fix if you had a window too large to be displayed at once in the window designer, the window structure would be resized to show only as much as the designer could show, leaving part of the window unavailable at runtime.

When you save a newly created file the save dialog now defaults to the name you set when creating the file. Good - I was getting a bit tired of having to type the name twice!

Window Previewer add support for setting Line Height for LIST/COMBO/DROP controls. Look for these under Tools | Options | Clarion | Window Previewer (Figure 7).



Window Previewer

Tab Control Style: Default

List Line Height: 0

Figure 7. Window previewer settings

You can use Ctrl-Tab and Ctrl-Shft-Tab to cycle between open files, the Quick View and the Dct Explorer in the Dictionary Editor. Handy!

You can configure if the locator in the data pad is below the toolbar, in the toolbar or not available via the Tools/Options/Clarion/General page.

You can now set the default characters, decimal places, justification and offset that the dictionary editor uses via Tools/Options/Clarion/Dictionary Editor Options/Column Options. Very nice.

You can get a trace of how the redirection system attempts to find a file from the File/Open File using Redirection File. This could be a big help for debugging problematic redirection files.

You can now close an application with the X in the top right hand corner of the tab.

You can now close the IDE if there are Apps open. I have to admit that over all the years, the one thing that has bugged me the most about Clarion is it's inability to graciously close down. I have never understood why I have to manually close an app before I can close the IDE. It just sticks in my craw, and I'd almost pay for the upgrade just to have this one issue fixed. And it's done in C8. Well, mostly. I have noticed that sometimes I have to try twice to close down the IDE; answering the "Do you want to save your changes" question doesn't necessarily result in the IDE closing the first time.

You can now use the redirection macro %Configuration% to get the currently active build configuration in your redirection files. I'm not sure I'd use this, as I'm far from a heavy user of redirection tricks, but it sounds interesting.

Improvements

Improvements are additions to existing functionality, not necessarily considered bug fixes.

AppGen now uses HTML Help.

Templates now use HTML Help (Template engine also still supports .HLP help). I've mostly stayed out of the discussions about help systems (although I've written a lot of articles, for some reason I've managed to avoid writing much in the way of help files), but HTML help replaced compiled help files a long, long time ago on Windows. It's nice to be at the party again.

Add hot key to the tool tip for the next and previous error buttons on the error pad. The more hot keys, the better.

Allow TRUE and FALSE equates as initial values for numeric variables.

Less chance of appearance of unexpected caption bar menu items on closing an MDI child window. When Robert Paresi asked about this in the newsgroups, Z replied with "It means that there is less of a chance of the appearance of unexpected caption bar menu items upon closing an MDI child window, IOW a workaround for the caption buttons drawn by the OS."

Show message with the column whose options are failing in the parsing. Cryptic.

Avoid extra redrawing of IMAGE controls on their resizing.

Changes

Changes, improvements, whatever. The change list in Clarion 8 includes:

.App and .Appx files are no longer added to the File category.

Added cleanup of the status message after the solution refreshed.

Avoid repeated calls to refresh the IDE statusbar with the same message. Some (though not all) users are reporting significant speed improvements in C8 as compared to C7.

Change to using HTML help in the templates.

Make sure the CWVersion property is set for Clarion versions on startup so 3rd parties can find it.

Relative paths in an {include} statement is now calculated relative to the redirection file rather than relative to the project directory. This makes it easier to have multiple levels of redirection files. More redirection magic!

"Refresh Embed Tree" options in the Application Options are removed.

Added template option for controls inside TABs to use COLOR:Window as the Background instead of TRN.

Allow locating of the #TEMPLATE's help file using redirection.

Better message when the OS UNICODE library isn't able to be loaded.

Better use of screen real-estate on Audit tabs.

Better handling of images on button controls.

Change text for Project Context Menu Delete Item to "Delete Permanently".

Change text for Project Context Menu Exclude Item to "Remove From Project".

If you click below the last element of a tree in the dictionary editor, now the last node is selected.

If an extension that is a Project/Solution/App type is open form the QuickOpen file it will open as a Solution using the Solution Dialog.

If value of ENTRY's USE variable has been changed after this ENTRY has received the focus but before drawing of changed value for the first time, selected text could be set incorrectly..

Key Editor now shows the KeyComponents in the first Tab instead of the 3rd, also the Comments were moved to the Audit tabControl.

Key properties general tab uses screen real-estate better.

QuickBrowse - changed some colors and font size.

Stop auto-display of the DataPad every time an .App is opened.

The dictionary editor now detects and removes corrupt key components and corrupt relationships. Scrubbing bad data is a good thing. It would be interesting to know just how many C6 issues are due to APP and DCT corruption. Probably a lot more than we realize.

When open a Solution/App/Project the extension in the Open Dialog now show sorted alphabetically.

When the DataPad can not display an item show a message with the Column/Table/Item with the problem .

You can now put the locators in the quick view in the dictionary editor in the toolbars or below the toolbars.

Fixes

These bug fixes are separate from the PTSS reports filed by users.

?A{PROP:Folllows} = ?B if ?B is a GROUP, OPTION, TAB or SHEET control could work incorrectly

Alias Tables were not removed from the application File Schema only from the Data Pad

Alias Tables were not added to the File Schema only to the Data Pad

Alias in a new procedure was sending to the Embed editor the columns from the Aliased Table and not the Alias Table. Before this was fixed, if you double-clicked on the alias you got the original file prefix, not the alias prefix. Easy to miss.

Alphas mapped to code in the range 128-191 occurred in string literals were translated to Unicode (in managed strings) incorrectly. If you had strings declared with initial values you could get some pretty odd results.

AppGen did not report errors on declaration of sections with duplicate names.

AppGen did not write Quick Code options to TXA files correctly

AppGen did not read Quick Code options from an .App file

*AppGen: Procedures Modified datestamp was not updated after applying changes in procedures local data made via Data Pad *note timestamp is updated when Procedure is saved, or if the edits to the Data Pad are done directly from the App Tree view (so the Procedure Properties is not open) the timestamp is updated when the Procedure loses focus*. I'm thinking this may have been at the heart of problems with data pad changes not being saved.

Back-space behavior was incorrect if typing mode is OVR.

Building PWEE information on invoking plain embed editor caused showing of "filled" icon for incorrect nodes in the Embed Tree.

Calling CREATE(file) for an MSSQL 2008 file that had a timestamp structure would fail.

Changeable data of the Generator instance could be accessed simultaneously from concurrent threads. I'm not sure what this means from my perspective as a user of the IDE, but it sure sounds like some thread-unsafe code which could cause App corruption or IDE instability.

Characters given by their codes in angle brackets in string literals could be parsed by designers incorrectly.

Closing an MDI child window would activate the first MDI window opened instead of the last.

Context help doesn't work with F1 in Embeditor (PWEE).

DCT Editor: Viewing the Controls tab for a column was prompting to save changes when another field was selected.

DCT Editor: The Control for a field could be reset to the default (or last saved state) when you selected the control tab. I've found this one pretty annoying, the few times I've come across it. Good to see it's fixed.

DCT: Auto Number was not displayed if you specified "Display Other Key Attributes" in the Dictionary Editor Options.

DCT: If you tried to insert a global variable in the middle of global data defined in the dictionary, then the field was displayed before the dictionary fields. When you reopened the app the field was correctly placed after dictionary variables.

DCT: Reading of a global variable from the dictionary derived from another dct's global variable can cause exception.

DCT: The "Derived From" list was not sorted based on the dictionary options.

DCT: The DCT Editor now displays <No Field Defined> when it encounters a key definition that has key components that point to a non-existent field instead of not displaying anything.

DCT: You could not change the file driver of a copied file without first closing and reopening the dictionary.

Debug data about extra fields in structures declared using parent structure might not be written to .OBJ file .

Disabled ENTRY placed on a TAB might not be drawn on activation of the TAB.

Disabled prompts could be drawn with incorrect background.

Disposed data could be involved into building the Embed Tree on the second entry to the dialog for a procedure.

Editing templates from the Template registry interface was not always opening the correct file.

Exception was thrown if the line that the error points at in the error pad was removed from the file and then you clicked on the error.

FORMAT and UNFORMAT could return incorrect value if picture is passed as a string or expression.

Generation could fail if there are orphaned embeds.

Global variables declared in the Dictionary displayed in the Expression Editor without prefixes.

If App Tree is in Modified View mode, the tree might not be refreshed after returning from the Procedure Edit dialog.

If Listbox's source or FORMAT string was changed in response to pressing VCR buttons, capture released incorrectly.

If the content of an ENTRY control is wider than the control, it could be displayed incorrectly before gaining focus for the first time.

Incorrect expansion of {n} in strings in TXA.

Initial values of string-like variables could be written to TXA file incorrectly if padded with '<0>' (for CSTRING) or space (for STRING and PSTRING).

Initial values of string-like variables could be read from/write to TXA/TXD files incorrectly if they contain <0> characters. Some significant TXA fixes there.

Newly added embed would not display in the embed tree if there are multiple #EMBED statements with the same ID but with different priority ranges.

Old apps could have invalid Picture in variables without the leading @ (now when the .App is opened those picture are corrected and stored back in the .App).

Old apps could have invalid options in variables that cause the Data Pad to crash. More data scrubbing - it's all good.

Output of properties with string values back to text could fail.

Parameters of attributes of WINDOW/APPLICATION/REPORT statement could be converted incorrectly.

Possible fault if no output to MAP file is specified in linker options.

Possible exception on reading TXAs with wrong values of the INITIAL attribute of variable declaration.

Possible failure if CREATEd items are added to the menubar before drawing it for the first time .

Possible exception on reading APP if name of base variable is misspelled or if APP has no dictionary.

Query to save dialog could be displayed for read-only applications.

REGION's frame could be redrawn incompletely after setting its color to COLOR:None.

Recovery file was not freed upon canceling the application.

Repeated calls to MATCH/REGULAR with malformed regular expression could cause a runtime exception.

STATIC/THREAD QUEUE based on local QUEUE could cause attempt to free some static memory during program shutdown.

Searching via the context menu in the Template editor would throw an exception unless you had done a standard find.

Setting the text of any control to something containing a apostrophe caused an error when the structure was saved.

Sometimes the Delete item menu in the project node deleted the selection even after clicking cancel.

Sometimes opening and (quickly) closing the app and re-opening another App can generate a NullReferenceException.

Statusbar showing last error was not cleared after the ErrorPad was empty.

Text Search sometimes gave a null result if the cursor was moved after a search.

The %TransactionManagerVirtuals #GROUP was defined twice.

The DCT editor would not generate the correct RADIO structures if you set the Values constraint list after setting the Choices list.

The DCT file would not store window and report controls >1024 characters in length.

The Derived From field was not read from a TXA file.

The Dictionary editor thew an exception on double-byte systems.

The FONT attribute was being automatically assigned as Sans Serif 10 by the designer if it was missing from the Window. And that could cause a lot of extra work if you deliberately left the FONT attribute off a bunch of windows.

The Listbox formatter built incorrect FORMAT string if columns have the header with alignment *(offset) equal to Left(2).*

The VALUE attribute of CHECK controls was not passed to the new application during conversion from previous version's application.

The WHERE condition on #RESTRICT statement in #CODE templates could be evaluated incorrectly.

The Window Previewer would not display a List that used a string rather than a queue as it's USE and used List styles.

The Window Previewer did not work if there was an IMAGE control with a second component of the USE attribute.

The area around themed push buttons blackened after numerous gaining and losing of focus .

The compiler reported an error if the @D picture token had a intellidate part.

The generated ship list had incorrect name for the Html help dll.

Typing in an ENTRY control with DECIMAL alignment could cause some unexpected behavior (position of caret, display of the value).

Unnecessary message box was displaying if the user canceled upgrading an app's dictionary to a newer version.

Value could be displayed incorrectly in ENTRY controls with @N pictures if some some digits should replace grouping characters.

Values of EQUATEs for standard icons and cursors was not always recognized by the Designer.

Windows OS problems with performing deferred move/size of controls on multiple resize events forced to be generated by changing of IDE layout.

You could try and set the Derived From field in an .App that has no Dictionary.

the InitialSize property had choices of Maximized and Iconized, the correct attributes are Maximize and Iconize (caused window parser/compiler errors).

Now, those aren't PTSS reports, those are internally found/fixed bugs and improvements.

Here are the fixed PTSS items:

PTSS reports fixed

PTSS 33603: Column Header Justification BUG

PTSS 34434: Context help doesn't work with F1 while highlighting Clarion language. In C7 highlighting a keyword and pressing F1 brings up the help, but not for that keyword. Brahn Partridge wrote an IDE addin to supply this functionality. I just tried it in C8 and it didn't work for me, but presumably it will soon, and perhaps does for others. In any case, even if you no longer need Brahn's help addin he has a bunch more you'll find useful in C8.

PTSS 35031: An unnecessary "Debug" message pops up after canceling DCT upgrade.

PTSS 36021: The window previewer could not display this window.

PTSS 36305: compiler error with intellidate picture.

PTSS 36360: Dropdown Listbox does not allow ' in it - Clarion 6 does.

PTSS 36932: Window previewer cannot display window structure.

PTSS 36962: Unable to preview attached window structure.

PTSS 36962: Unable to preview attached window structure.

PTSS 36963: Disabled PROMPTs are displayed incorrectly.

PTSS 36986: Some list boxes have invalid format after the conversion C6.3->C7.2.

PTSS 36987: REMOVE() function not working correctly.

PTSS 36995: Buttons over not redrawn properly on MAXed screen.

PTSS 36997: The window previewer cannot open some windows after the conversion C6.3->C7.2.

PTSS 37028: Numbers disappear while entering in entry field.

PTSS 37036: Attached Project crashes the Linker.

PTSS 37037: PTSS #37027 still not working.... {PROP:FOLLOWS}.

PTSS 37041: gpf using queue(someotherqueue),static.

PTSS 37047: Dissappearing Frame Menu on Maximize/Restore sequence when MDI Child is maximized.

PTSS 37056: Drop Combo Not Working Correctly adding a new entry.

PTSS 37081: Report designer changes characters entered in Job name.

*PTSS 37098: Expression editor does not show prefixes of *Global* DCT Tables.*

PTSS 37118: Read Only apps prompt to save.

PTSS 37130: PROP:COLOR on Region does not work properly.

PTSS 37137: Checkbox VALUE('X','') conversion error.

PTSS 37150: Fields do not resize properly when changing page layout...

PTSS 37181: Not updating last modified date.

PTSS 37187: Window Preview - doesn't display.

PTSS 37195: POPUP menu items lost.

PTSS 37198: Disabled Entry Loses Text.

PTSS 37201: Problems with seeing static classes in Debugger.

PTSS 37236: Loosing the File-Browsing List Box files.

PTSS 37261: Unable to change driver on copied table.

PTSS 37273: Failure on #RESTRICT in #CODE Template.

PTSS 37355: Left mouse button lockup.

PTSS 37365: Window Hangs on open.

PTSS 37398: Match given certain regex patterns can cause an Exception.

PTSS 37464: Wrong parsing of procedure parameter prototypes when using CONST.

PTSS 37717: Data Pad is empty.

PTSS 37748: DCT: The Foreign Key and Public Key label text does not switch when loading an existing Many:1 relationship.

PTSS 37792: PWEE file needs a unique temporary name. If you had multiple developers working in the same shared directory, you stood a good chance of getting a file lock problem on the temporary embeditor file. This file name is now unique to each process.

Workarounds

Windows paints semi-transparent area around themed button incorrectly in response to the WM_PRINT message if height of button less than 18 pixels

Do not allow generation to fail on malformed relations in the dictionary

And the verdict is...

There are a total of 216 features, fixes, improvements and changes listed in the readme for the first release of Clarion 8. Doubtless there will be many more as the beta progresses.

One of the concerns I had after the preview at last fall's ClarionLive DevCon was the lack of any really dramatic new feature in C8. And I still don't know that there's one absolutely killer new "gotta have it" feature in this release. But there are a number of mid-sized items that make for a pretty decent package. These include:

- IDE visual improvements (some are being hotly debated, as you'd expect)
- IDE speed improvements (not touted as part of the release, but being reported by some users)
- Locators everywhere in the IDE
- Remembering of embed list layout
- Better hot key support
- Slider control in the toolbox
- TXA editing of data pad items
- Three state checkbox

- Runtime visual improvements (gradients, better appearance of disabled icons and images)
- HTML help support
- Redirection system improvements
- The IDE settings cleanup tool

Add in the numerous minor improvements, changes and bug fixes and I think C8 is a pretty compelling package.

About those bug fixes

I expect to hear a certain amount of complaining about bug fixes showing up in C8 and not in C7. I usually hear the argument phrased this way: "I paid for C7, and it still has bugs. Why should I pay again to have those bugs fixed?"

There are few developers who ship absolutely bug free code, and none I would guess who ship products as complex as the Clarion IDE. Yes, there are bugs in C7, as there are in C6 and in every version of Clarion I've ever used. If programmers only got paid for shipping bug-free software, almost all of us would be out of business.

For the gun-shy among us

The migration from C6 to C7 was a painful experience for many. There were templates that wouldn't register, APPs and DCTs that wouldn't import, third party suppliers who needed to be cajoled into supporting the new version, teething pains for the updated runtime library, and so on.

The pain of that platform shift is largely behind us now. All the templates, dictionaries and applications that work in C7 will, by all accounts, work without difficulty in C8. I expect the move from C7 to C8 to be more along the lines of the move from, say, C5.5 to C6. There are improvements, but the foundation is the same.

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 Dave Harms on May 2 2011 (comment link)

For those unhappy with the large icons, Brahn Patridge has an addin that will let you use smaller icons on the main toolbar.

by Dave Harms on May 2 2011 (comment link)

There are some issues with toolbar icons not displaying properly. Arnold Young points out this workaround from the newsgroups:

> Go to Global Properties, App Settings, App Manifest, Check " do not use themed controls"
> -- Seems to fix VCR buttons on browses.

by Dave Harms on May 2 2011 (comment link)

There are some reports of IDE freezes. Doesn't appear to be a widespread problem. It is being looked into.

by Dave Harms on May 2 2011 (comment link)

Robert Paresi has noted a regression with 32x32 true color icons . Place it on a button, flat, with left justified text. Hover the mouse over the running app and the icon becomes squished. He's provided a sample app showing the problem. Diego quoted a note from the Axialis icon editor docs:

> "XP with Alpha Chanel images are displayed correctly in Windows XP and later only." "True Colors (RGB - 24 bits) is not standard icon format" "To create RGB icons we recomend you choose XP with Alpha Chanel (RGB/A 32 bits)"

by Dave Harms on May 2 2011 (comment link)

Duplicate Addin problems? Note from Brahn:

> To cleanup all your addins at once try removing (or just moving to a backup location!) this folder : %appdata%\SoftVelocity\Clarion\8.0\Addins
>
> Any addin that you install using the AddinManager goes into that location.

by Dave Harms on May 2 2011 (comment link)

Mike Hanson has posted a template that causes the C8 AppGen to lock up.

The problem is with calling a #Group in the extension's description attribute. Mike uses this technique in many SuperTemplates which cannot be used with C8 until this is fixed.

by Dave Harms on May 2 2011 (comment link)

Randy Rogers has also posted a template system bug that causes a crash.

by Dave Harms on May 2 2011 (comment link)

Z is reporting that Mike Hanson's template bug has been fixed and an update will be released Monday or Tuesday.

by Dave Harms on May 2 2011 (comment link)

Note: If you're compiling with C7, in the C8 IDE, the IDE will copy in the C8 runtime DLLs instead of the C7 runtime DLLs. You'll have to copy the C7 runtime DLLs in manually, and you may want to disable the automatic copying of the DLLs thereafter (at least until this issue is fixed).

by Dave Harms on May 3 2011 (comment link)

Some clarifications from Bob Z regarding C7 and C8 runtime DLLs:

- Third party DLLs do not have to be rebuilt for 8.x apps unless those DLLs are local linked
- You can run 7.3 executables using the 8.0 runtime DLLs
- You may not be able to do the reverse, however; if the C8 executable uses a C8 runtime feature such as the slider control or three state checkbox, the app will GPF because the 7.3 runtime doesn't support those features.
- If you do get a crash using the C8 runtime DLLs with a 7.3 app SV would like to know about it.

by Dave Harms on May 3 2011 (comment link)

There have been several reports of problems with WMF images. Lee White has posted PTSS 37928.

by ClarionMag user on May 3 2011 (comment link)

Build 8247 is available.

Release Notes: C8.0.8274

Fixes/Features/Changes

FEATURE: Support icons and bitmaps that use an alpha channel. (images with alpha channel are not supported in reports and in IMAGE procedure until switch to EMFs)

CHANGE: Speed up the conversion of C7 DCT files to C8 DCT files CHANGE: Adjust Tabs/Text/Image spacing in Pad Tabs and Document Tab CHANGE: Datapad: Change the Edit Data as text icon CHANGE: Order of cached embed responses / code generation of structures to match C73 CHANGE: Support greyscaling images with alpha channel

FIX: Displaying of metafiles FIX: Icons in Pad tabs were not showing for certain DPI/Resolution combinations FIX: Icons on buttons could be stretched FIX: If any Pad is pinned on the left side of the App and unpinned sometimes the App Tree view was not resizing FIX: Mis-aligned icons on VCR buttons on the LIST control FIX: PROP:Checked returned '0' instead of '' for normal check boxes if they are in the unchecked state FIX: Selecting global Data in the DataPad was changing the Font FIX: The %BIN% redirection macro was being expanded to the directory where the C8 binaries where located rather than the directory where the binaries for the version of Clarion in use where located FIX: When App's are set to auto-open and there area other files in the Solution the App document might not refresh FIX: When creating a new Clarion 7.3 Clarion version within C8 the list of compilers was not set correctly FIX: add *.chm to the default red file

PTSS 37919: Clarion 8 Icon Issue when hovering PTSS 37923: AppGen lockup if the description of a #CONTROL/#EXTENSION template used a #GROUP returning a value PTSS 37924: Second Icon issue in Clarion 8 PTSS 37928: C8 - Vector image positioning is broken PTSS 37933: Prop:Check / Prop:Checked misbehave

by Bruce Johnson on May 4 2011 (comment link)

> -- I expect to hear a certain amount of complaining about bug fixes showing up in C8 and

> not in C7. I usually hear the argument phrased this way: "I paid for C7, and it still has bugs. Why should I pay again to have those bugs fixed?"

Actually, you don't. Well most people don't. Included in the C7 CSP is the Clarion 8 beta's and Gold release. So anyone on a C7 CSP gets these bug fixes without paying more cash.

Those on an SMP of course do not have access to these fixes, but my guess is that these are in the minority.

If you are on neither the CSP nor SMP then you wouldn't get the fixes even if they were released as part of C7.

---

by Rhys Daniell on May 4 2011 (comment link)

Lipstick on the pig.

Although upgrading from c6 would be nice, I don't see any productivity or feature improvements which justify the very substantial effort which would be required in our little shop to migrate and test all our applications from C6 to C7.

Fwiw it turns out running VirtualPC under Win7/64bit takes most of the pain out of using the creaky C6 IDE.

Leaving aside the elephant in the room (.Net support) the things that we long for in Clarion around here are better support for SQL and XML.

Surely by now most Clarion developers have moved to an SQL back end or are taking a serious look at it. Clarion's unique data dictionary lends itself to not only tight integration with SQL table and view declarations but also support for stored procedure and user defined function development. And, while moving the IDE and dictionary to a new platform (SharpDevelop) why couldn't SQL have been used as the repository? It would have offered improvements in stability and performance and an easy path for 3rd party tool developers.

Similarly, if your application does any sort of communication with other systems, XML has become a fact of life. While iQXML and xFiles do quite a good job of XML support, SV could give us a tool that takes things to a new level.

Instead, what we see in Clarion releases for the last couple or so years are minor bug fixes and a few end user cosmetics. They forget that most of us are in the business of developing business applications, where the end user cares less about cosmetics and more about functionality, reliablity, and ease of use.

After considerable thought, and much debate with fellow developers, I just don't see a business case for moving to these latest Clarion versions. If you are a dilettante developer with time and money to spare, go ahead. But on a cost-benefit basis an 'upgrade' is impossible to justify.

Just sayin'

---

by John Morter on May 5 2011 (comment link)

Hi Rhys,

I find myself in (reluctant) agreement with you regarding lack of real benefit in moving to C7/C8 ... That's "reluctant" 'cos I'd truly like to have a good reason to make the move - but, try as I might, I keep being driven backwards by the many annoyances I encounter in the new IDE - with nothing compelling enuff to force me over the hump !

Just as one for-example: The Embed/Filled button still doesn't work properly (I'll try reporting that to SV again - for the 3rd time).

After my disappointment with C7, I was looking forward to C8 with expectation of it being a "major advance" - and I find instead that it probably should have been called just C7.4

So, for the time-being, I'll stick with my C6.78 environment - Yep, that's C6 with the C7.3/C8 Templates ('cos the Templates are the killer-feature of Clarion, for me, and I wanna be sure I'm using the latest instance of 'em) ... It took a bit of tinkering to get it all working - but the result suits me just fine (where all my work is with pure-ABC [no "raw SQL"] over a MS-SQL back-end).

I don't want this to sound totally negative, tho ... I'll keep supporting SV by maintaining status of my Core Subscription - 'cos Clarion is still the best App.Dev. environment around - and I do want SV to continue striving.

Regards, JohnM

---

by ClarionMag user on May 5 2011 (comment link)

Rhys &amp; John,

I've observed the C7/8 cost/benefit analysis from several perspectives now. I gather a lot of feedback through Clarion Magazine. I use the new product in my own work. And I work on a very large application with a number of other Clarion developers.

There is no universal agreement that moving to C7/8 is a net benefit. On that large project I mentioned, there are team members who love C7, some who tolerate it, and one who frequently expresses hostility towards it.

I believe the team as a whole is far better off with C7, even though some team members are still experiencing a significant learning curve. Build times are much faster, downtime is less, and for those of us doing any amount of hand coding C7 is far, far superior. (And I maintain that for large systems, the stability and flexibility of the system is directly related to how much of the business logic exists in testable, reusable hand code rather than being buried in embed points.)

To me, that team seems a lot like the C7 community in microcosm.

John, as to your 7.4 point, SoftVelocity seems to be better at incremental releases than at punctuated releases. Z has said they are considering an annual subscription model, which I suspect would satisfy many of the current users and would remove the issues around what constitutes a major release.

---

by Bruce Johnson on May 5 2011 (comment link)

Hi Rhys,

"Although upgrading from c6 would be nice, I don't see any productivity or feature improvements which justify the very substantial effort which would be required in our little shop to migrate and test all our applications from C6 to C7."

I hear this argument regularly. From people using Clarion 5.5, Clarion 5, Clarion 4, or in one epic case Clarion 2 for windows.

The bottom line is that the new IDE has numerous benefits that improve productivity on a daily basis. It's faster to generate and compile for starters - the editor is better and so on. C8 has built on that improving it even more.

As with most Clarion updates though the benefits are really only experienced when you get to use the tool day by day. Also, unlike most updates, C7 contains a lot of brand new code, so there have been a lot of issues with bugs etc. That has been steadily improving, and seems to be even better in C8.

Unlike earlier updates there's also a definite learning curve in using the new tool. So it takes a bit of perseverance to get going. That's life I guess.

Of course a sizable fraction of users will never make the leap - just like some never made the leap to cw 4, or to ABC. That's ok - it's perhaps Clarion's biggest compliment that we can continue to create apps in C6, almost 10 years after C6 first shipped, which are still cutting-edge in the market today. Certainly there's nothing in C6 that holds you back so clearly it's both a viable, and workable alternative.

Cheers Bruce

BACK TO TOP