Clarion Magazine

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

Clarion News

Read 12 Clarion news items.

Articles

Tip of the Week: EXP Files Made Easy

November 9 2011

Creating export files by hand can seem bizarrely complex. Unless you let the compiler create the export statements for you.

Tip of the Week: The Type Finder

November 16 2011

There are lots of nifty navigation aids in the C7/8 IDE including the type finder.

Tip of the Week: Another Way To Open Files

November 23 2011

Sometimes opening a file is as easy as highlighting text.

Displaying and Interacting With HTML Pages

November 25 2011

It's not that difficult to display HTML pages inside a Clarion application. Andrew Popoff takes things a step further and shows how to respond to clicks in those pages with Clarion code.

Set And Forget DLL And LINK Attributes For Class Libraries

November 27 2011

Few things drive David Harms around the bend faster than trying to remember how to set DLL and LINK mode defines for applications that use classes exported from another DLL. Here's his solution that only requires one project define, in just one app, ever.

Populating Lists From Queues In Clarion 7 / 8

November 28 2011

In all versions of Clarion for Windows, up to Clarion 7, creating a list box from a queue was easy. The process is different in Clarion 7/8, but it's actually easier.

Thread-Safe Runtime Translation With The In-Memory Driver

November 30 2011

As of Clarion 6, global queues have some risk attached to their use thanks to the runtime's ability to use true threads. The In-Memory Database Driver is a popular replacement for queues thanks to its thread safety, as Nardus Swanevelder shows in this update to his previous article on runtime translation.

Passing Queues

November 30 2011

Having recently conquered passing GROUPs not only between procedures but between threads, there remains one more complex structure. Dr. Parker's new Everest is Queues (without concern as to passing them across threads).

Tip of the Week: Defeating Circular Dependencies

November 30 2011

No, this isn't about how to remove circular dependencies (though you should); it's about how to compile source only when you have circular dependencies.l

Custom Methods And Derived Classes

November 30 2011

Nardus Swanevelder explores the often-ignored world of template-generated derived classes.

Clarion News

ProArchive Demo

An installable demo of ProArchive is now available. This is one of the demo applications that ships with the ProArchive template. It demonstrates the following: Archiving parent and related child records; Archiving records with Blob fields; Archiving records with Memo fields; UnArchiving records; Using a Process Template to archive multiple records; Using Handcode to archive records; Using the optional parameter to open an existing browse directly to the archive file; Toggling a browse between the active file and the archive file; Using the same report to display records from either the active file or the archive file. Both the installer and the demo application are code-signed. The installer creates a shortcut in Programs to run the demo app and by default the app installs to C:\ProArchive Demo.

Posted November 8 2011 (permanent link)

It's Capevember

It's Capevember this month, because CapeSoft is sponsoring ClarionLive! with a GotoMeeting licence this month. This means that practically everyone who'd like to can attend the ClarionLive! sessions during Capevember. On November 4 Geoff did a session "Everything you need to know about security in your application". On November 11 Bruce will do a session on graphing, where he'll be showing you Graphing techniques you can use in your applications. And the week after (that's the 18 November), he'll be introducing NetTalk 6 and showing you some of the features he's been working on the last couple of months, that you can utilize in your web server applications. Rob (if you haven't been introduced to Rob yet, he's the other partner in CapeSoft) will do a session on writing reports at runtime using RightReports on 25 November. The sessions will be recorded for those who are unable to attend (due to time differences or whatever). There will also be some discounts for those who attend, so you'll want to be there to get the most out of the sessions. Don't forget to wear your CapeSoft T-shirt. CapeSoft T-Shirt wearers have been known to win iPads.

Posted November 8 2011 (permanent link)

Capevember DevCon Special

During the month of Capevember, CapeSoft has put together a deal with ClarionLive for those who missed the Clarion International Devcon. You get the entire set of CIDC 2011 talks (that's three days of some of the best names in Clarion - Bob Zaunere, Diego Borojovich, Dave Harms, Pierre Tremblay, Bruce Johnson, Shawn Mason, Mike Hanson, Bob Foreman, Rick Martin and Andy Wilton) on streaming video for 30% off the normal $497 price tag. That's only $347, and they'll toss in all the workshop sessions that were done during the 3 days after the CIDC. So you get about $694 worth of goodies for only $347. And you have 90 days to chew through the material. And if you're not 100% convinced after 90 days that the 50 hours of the best Clarion has to offer is worth at least $347, then mail CapeSoft, and they'll refund you the full $347. And they'll still give you access to the downloads. And if you missed out on the CLDC in 2010, you can grab those when you get your CIDC 2011 videos

for an 80% discount. That's down from $395 to a just $75.

Posted November 8 2011 (permanent link)

CalendarPro Wrapper Template 2.12

Version 2.12 of the CalendarPro Wrapper Template is now available for download. This update includes: New 'Disable at Runtime' option; New 'NewDate' Event Trigger Effect option; New 'GetEventIDViaRecID' method; A new example has been added to the demo app to demonstrate populating and navigating the Calendar via a standard Clarion Browse; Fix for incorrect selected date values returned to the Calendar / DatePicker 'NewDate' methods. 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 November 8 2011 (permanent link)

Codejock Discount

According to Noyantis' special agreement with Codejock software the 30% discount is good towards all ActiveX components and renewal subscriptions through the end of 2011. To use it select a product, add to cart. After that use Coupon Code (enter it into field in Cart form) CLARION and push Update/Recalculate button. You will see a new discounted price (and small (E) icon)!. After you can check out.

Posted November 8 2011 (permanent link)

Stimulsoft 20% Discount

A 20% discount is available for Clarion developers who want to buy Stimulsoft Reports.Net, Stimulsoft Reports.Web, or Stimulsoft Reports Designer.Web. The following types of licenses are included in this offer: Single License - 1 developer; Team License - up to 4 developers; Site License - unlimited number of developers to one physical address. Select a product(s), add to cart. After that use Coupon Code (enter it into field in Cart form) CLARIONSTIMUL and push Update/Recalculate button. You will see a new discounted price (and small (E) icon)!. After you can checkout.

Posted November 8 2011 (permanent link)

SetupBuilder WISE Crossgrade Offer Ends Nov 30

WISE has officially retired. If you are concerned your WISE product doesn't support your software deployment requirements, consider a jump to SetupBuilder Developer Edition. For users who want to move over from WISE to SetupBuilder, there is currently a special offer: switch to SetupBuilder 7 Developer Edition including a full 1-year maintenance and support subscription plan and pay $245 instead of $399. This special offer expires November 30, 2011. Contact sales@lindersoft.com to save 40% off your move to SetupBuilder Developer Edition. Proof of ownership required.

Posted November 25 2011 (permanent link)

FullRecord 3.04

FullRecord 3.04 includes fixes to temporary files being generated in the working folder rather than in the OS folder. There is also now an option to use the IMDD (not included) to generate the temporary file in memory.

Posted November 25 2011 (permanent link)

MainToolbarExtras AddIn Updated

MainToolbarExtras 0.11, 17 automatically re-opens the StartPage when the solution is closed. This feature is enabled by default; use the context menu on the main toolbar to disable it.

Posted November 25 2011 (permanent link)

SkinFramework Wrapper Template 2.07

Version 2.07 of the SkinFramework Wrapper Template is now available for download. This update includes: Codejock v15.1.0 -> v15.1.3 compatibility added; C55 Support Code Removed; Office 2010 Skin added; Safari Skin added; New Option : 'Hide AppFrame During Close Down'; New 'Disable at Runtime' option. 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 November 25 2011 (permanent link)

DMC 2.3.3.1539 HotFix

This release is available, free of charge, to all DMC customers who have an active DMC maintenance and support plan subscription (and to all evaluation level users). Changes include: Corrected a bug during updates done in data transfers on compound keys; Corrected a bug during updates done in data transfers when the column names contains a prefix.

Posted November 25 2011 (permanent link)

SoftVelocity Clarion.NET AppGen Demo

SoftVelocity will be on Clarion Live on November 30 at 9-11am (Pacific Time) doing a .Net AppGen presentation, and again on December 16th. Topics for 11/30: An Introduction to AppGen.Net - covering the .Net template registry, WinForm app and WebForm; Wizard and code generation, the object datasource data access layer shared by winform/webform, working in AppGen using the current templates both Winform and Webform, writing, editing and debugging the .Net T4 templates and (time permitting) using Subversion to get updates. (Presenters: Robert, Diego and Pierre) The presentation is recorded and will be available for download. A new Clarion.Net release (with the new .Net AppGen) will be available the same day after the webinar or the day after.

Posted November 30 2011 (permanent link)

The ClarionMag Blog

Tip of the Week: EXP Files Made Easy

By David Harms

Posted November 9 2011

Most of the classes I write have no connection to the ABC class library. They don't have an `!ABCIncludeFile` comment, and they don't derive from or extend any ABC classes. And quite often I've written those classes with this kind of header:

```
CM_Demo_Invoice CLASS,TYPE,MODULE('CM_Demo_Invoice.CLW'),LINK('CM_Demo_Invoice.CLW')
```

A class declared this way will be linked into every DLL or EXE that uses it. And for most of the classes I write, that's perfectly acceptable.

The alternative is to export the classes from a base data or classes-only DLL (which, incidentally, I have begun doing). But when you do that you have a problem. You need to export the "mangled" class and its public methods from the DLL.

I don't know about you, but I've always found name mangling a bit of a hassle. There are some tools available. For instance, the Pro2EXP sample program that ships with Clarion will create manged procedure prototypes for you (but it won't do classes).

Fortunately, Ilka Ferrett told me how she handles this, and it's a really easy technique. Later I discovered that Carl Barnes also mentioned this in a comment to Jeff Slarve's article on creating a DLL for your code library, and gives credit to Chris Buechler for the idea.

If you've ever had to hand code an EXP file, or you've used the "inside the EXP file" embed to add an export to a DLL, this will all make instant sense to you. The process goes like this:

1. Write the class and compile it into the DLL
2. Write some code that uses the class in the DLL
3. Compile that code and get an error message
4. Grab the relevant bits from the compiler message and paste them into the EXP file (or EXP embed), adding @? at the end (with a space before the @).
5. Recompile the classes DLL
6. Recompile your code that uses the classes DLL

For instance, let's say I've added a new method called `ReplaceWord` to Rick Martin's string class. I've exported that class already from my classes DLL, but when I write some code outside that DLL that attempts to use `ReplaceWord`, I'll get an error message:

```
Unresolved External REPLACEWORD@F16CM_SYSTEM_STRINGsbsbOl
  in CM_System_String_Tests003.obj
```

All I have to do is copy `REPLACEWORD@F16CM_SYSTEM_STRINGsbsbOl` onto a new line in my class library EXP file and add a `@?`:

```
  REPLACEWORD@F16CM_SYSTEM_STRINGsbsbOl @?
```

I compile the DLL, the method is exported, and my code that uses the DLL will compile and link. Just

remember that @? on the end or the export doesn't happen.

The downside to this approach is that you don't know what to export until such time as you actually try to use a class or method. But it's quick and it's easy.

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 David Hlavac on November 30 2011 (comment link)

Hi David, Actually I use the "Class: Object Exporter" from the ABCFree templates to do this. It's easy and also exports the object and VMT.

BACK TO TOP

Tip of the Week: The Type Finder

By David Harms

Posted November 16 2011

The next time you have an app open, press Ctrl-T from just about anywhere in the IDE. You'll see a dialog box with an Enter type name prompt. If, for instance, you type "br" at that prompt you'll see a list of classes that begin with those letters. You can select one of the files and it will be loaded into the IDE.

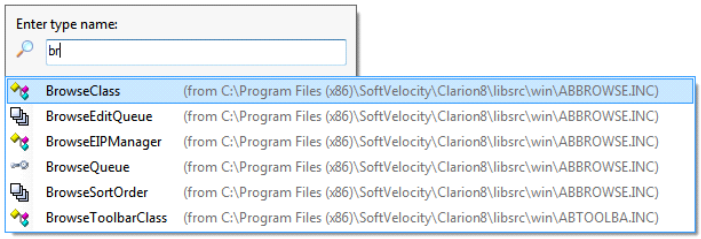


Figure 1. The type finder

The type finder isn't comprehensive, at least in my experience. It seems to be restricted to globally available types; if you have a typed data structure of class that's local to a procedure, it won't show up in the list. But if you're using your own globally availalable TYPEd classes you can expect to see those in the list.

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: Another Way To Open Files

By David Harms

Posted November 23 2011

Last week I showed how to use the type finder to bring up class INC files. But what if the INC isn't really what you're after? What if you've brought up ABBROWSE.INC and you want to look at the source for the `BrowseClass.ApplyRange` method?

I don't know if there's a really fast way to do that, but you can open ABBROWSE.CLW fairly quickly. Just highlight that text and right-click for the context menu.

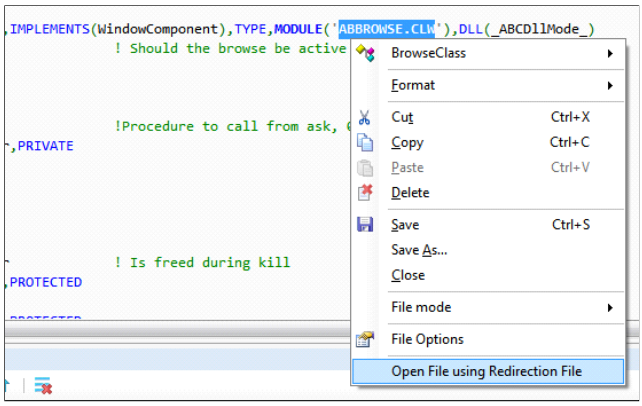


Figure 1. Opening the method source file

Choose Open File using Redirection File and the IDE will locate and open the file.

This Open File technique isn't restricted to class files; you can use it on any selected text.

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

Displaying and Interacting With HTML Pages

By Andrew Popoff

Posted November 25 2011

HTML is a simple markup language for documents displayed in web browsers. You can use it together with Cascading Style Sheets (CSS) and JavaScript to create interactive pages and pages with original design, and display these pages inside your applications.

It isn't difficult to show html pages in a Clarion app. For this purpose you can use the `Shell.Explorer.2` OLE control. This control is on all computers that have the Internet Explorer browser installed. By default, IE is installed on all computers running Windows, so this control is always available.

To open a web page with the OLE control you execute the following code:

```
?OLE{'Navigate("file://' & CLIP(inFileName)  & '")'}
```

Links are a part of almost every HTML document, and the main task is to intercept the click on the link in html document and perform the action. You do this by registering a procedure to handle the OLE control's events:

```
OCXREGISTEREVENTPROC(?OLE, MyOleEventFunc)
```

The `MyOleEventFunc` procedure (or whatever you decide to call it) has a certain prototype:

```
MyOleEventFunc PROCEDURE(*SHORT inReference, SIGNED  inOleControl, LONG inCurrentEvent),LONG
```

The Clarion 6 help says this about the parameters the procedure receives from the operating system:

> ***SHORT** A Reference parameter to pass onto the following other OCX library procedures: OCXGETPARAM, OCXGETPARAMCOUNT, and OCXSETPARAM as their first parameter.
>
> **SIGNED** The field number for the control. This is the same number that is represented by the control's field equate label.
>
> **LONG** The number of the .OCX event. Equates for some pre-defined event numbers are contained in the OCXEVENT.CLW file.
>
> The LONG return value indicates to the operating system whether any further processing is necessary. Returning zero (0) indicates some further processing is necessary (like updating a USE variable or unchecking a radio button), while returning any other value indicates processing is complete.
>
> Processing the events generated by an .OCX control must occur quickly, since some events have critical timing. Therefore, there should be no user interaction possible within this procedure (such as WINDOWs, ASK statements, or MESSAGE procedures). The code should process only what it needs to, just as quickly as possible (usually, this means eliminating all mouse events).

Consider the following HTML page:

```
<html>
  <body>
    <a  href="CLICK://I_WANT_TO_CATCH_THIS_LINK">Try me</a>
  </body>
</html>
```

Here's the code that executes when the user clicks on the link:

```
MyOleEventFunc      PROCEDURE(*SHORT inReference, SIGNED inOleControl, LONG inCurrentEvent) ! ,
loc:Event       CSTRING(1024)
loc:Link        CSTRING(1024)
 CODE
```

```
IF inCurrentEvent <> OCXEVENT:MouseMove

   loc:Event = inOleControl{PROP:LastEventName}

   IF CLIP(loc:Event) = 'BeforeNavigate2'
      loc:Link = OCXGETPARAM(inReference, 2)

      IF SUB(UPPER(loc:Link),1,8) = 'CLICK://'
         loc:Link = SUB(loc:Link, 9, LEN(loc:Link) - 9) ! cut the 'CLICK://'
         IF UPPER(loc:Link) = 'I_WANNA_CATCH_THIS_LINK'
            MESSAGE('CATCH IT')
         END
         OCXSETPARAM(inReference, 7, 1) ! cancel navigate
      END
   END
END
RETURN TRUE
```

And that's all you need to display and process HTML pages in Clarion.

You can find the code for this in the SimpleHTML.zip in the downloadable source.

Complications

All my life I have dreamed about a desktop that looked like Figure 1.

Figure 1

In a regular desktop application the main problem is the correct displaying of action icons when the window resizes.

Figure 2

This task is quite cumbersome in Clarion but is easily implemented in HTML.

In the downloadable source zip you'll find the `CFCHTMLClass` (cfchtml.inc, cfchtml.clw) This class displays HTML documents and handles the clicks on links. Besides these actions, this class solves two problems familiar to me.

First of all, control `Shell.Explorer.2` has the standard IE context menu. I want to block this menu.

Second, when the focus is inside the OLE control, keys like Esc, Ctrl+F4, Ctrl+F6, F10 and others are processed by control, not by Clarion's runtime. Accordingly, it is necessary to intercept these keys and pass them to Clarion's `ACCEPT` loop. If you look at the window handles *after* downloading the HTML document you will see the following:

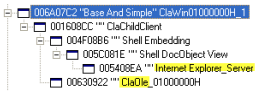


Figure 3. Windows in use

The Clarion OLE control and the Internet Explorer Server class window are marked by yellow. All displaying and processing of events occurs in the Internet Explorer_Server window.

At the same time you can see from the picture that OLE control is not hierarchically connected with Internet Explorer_Server window.

To intercep the context menu and clicks on the service keys it is necessary to intercept the `WM_RBUTTONUP`, `WM_CONTEXTMENU`, `WM_KEYDOWN`, and `WM_KEYUP` events coming to the Internet Explorer_Server window. The way to do that is to to subclass the window. Since it's possible to put multiple HTML documents on a window, each of those document windows needs to be subclassed. One more time please note that the Internet `Explorer_Server` window appears dynamically only after the HTML document has been downloaded.

To subclass I have to find all windows with class of Internet `Explorer_Server`. This is achieved the `EnumChildWindows` and `GetClassName` API calls. I need to avoid repeated subclassing, so I save all the windows to a queue. Then I go through the queue and subclass the every window. Also I've set the flag that window is already subclassed. When I go through the queue the second time I will check that flag. If it `TRUE` then I skip that window.

The `CFCWndProc` class (CFCWndProc.inc, CFCWndProc.clw) that I use for subclassing is beyond the scope of this article.

Since there is only one global class doing the subclassing I need to frame potentially thread-unsafe actions using a critical section. You can see the code which implements all the above in the file cfchtml.clw; look for the `CFCHTMLManagerClass.SetIEsCallback` method and the `CFC_HTMLEnumerateWindows` procedure.

The subclassing procedure is simple enough:

```
CFCHTMLCallbackClass.CallBackProc PROCEDURE(ULONG hWnd,ULONG inMsg,ULONG wParam,LONG lParam)
loc:WM_RBUTTONUP         EQUATE(0205h)
loc:WM_CONTEXTMENU       EQUATE(07Bh)
loc:WM_KEYDOWN           EQUATE(0100h)
loc:WM_KEYUP             EQUATE(0101h)
 CODE
 CASE inMsg
 OF loc:WM_RBUTTONUP OROF loc:WM_CONTEXTMENU
    RETURN FALSE
 OF loc:WM_KEYDOWN
    CFC_PostMessage(0{PROP:Handle},loc:WM_KEYDOWN,wParam,lParam)
    RETURN FALSE
 OF loc:WM_KEYUP
    CFC_PostMessage(0{PROP:Handle},loc:WM_KEYUP,wParam,lParam)
    RETURN FALSE
 END

 RETURN PARENT.CallBackProc(hWnd,inMsg,wParam,lParam)
```

The remaining details

`Shell.Explorer.2` has the same properties which are mentioned at the browser options. It means that if picture displaying in IE is off they will be disabled in their own window. It is necessary to take this into consideration when loading HTML documents and probably to warn the user about this setting.

The PNG format is more desirable for picture displaying, and support for this format was made standard in IE version 6. Also remember that the images referenced in your documents must be available to the end-user.

When you click the default link you may hear a sound effect. You can turn this off in the Internet Explorer preferences.

There are some additional methods (`IsImagesEnabled`, `GetIEVersion` and `DisableClickSound`) which you can use to ensure that image display and sound effects work as expected.

Finally, if you wish you can create an HTMLdocument dynamically (it's just a text document) and then load it into the OLE control.

Download the source

Andrew Popoff is a Russian developer who has worked with Clarion since 2000. Previously with SealSoft, Andrew develops corporate applications, and is also the developer of the third party products xXPFrame, xXPPpopup and xReportPreview. He writes a blog at news.clarionlife.net and is the content manager of the Russian Clarion resource clarionlife.net. He occasionally writes Clarion articles for the Russian developer community.

Article comments

BACK TO TOP

Set And Forget DLL And LINK Attributes For Class Libraries

By David Harms

Posted November 27 2011

Classes have completely taken over my Clarion programming. Yes, I still create procedures using AppGen, and I still (reluctantly) write some logic in embeds. But for the most part, I write classes.

There are fundamentally two ways an app can use a class. One is for the class to be compiled into the app, and the other is for the class to be compiled into a DLL that can be used by any number of apps. This DLL could be your base DLL, or it could be a completely separate DLL. For my own generic classes, those that aren't tied to any specific application, I create a separate classes DLL (and I'll show how to do that a little later).

The disadvantage of the first approach is that if you need to make a change to the class, say a bug fix to one of the methods, you need to recompile every app that uses that class. The advantage is it's hard to GPF your app using this technique.

The disadvantage of the second approach is that you have to add some extra information to the class definition, and you have to add some extra project defines to each and every APP (or hand coded project) that uses the class DLL, and if you get any of this wrong your app will almost surely GPF. The advantage is that if you make change to the class that doesn't affect any of is existing exports, you don't need to recompile all the apps that use that class. You just drop in the new DLL.

An ABC example

Here's a typical ABC class header:

```
StepClass     CLASS,MODULE('ABBROWSE.CLW'),TYPE|
                ,LINK('ABBROWSE.CLW',_ABCLinkMode_),DLL(_ABCDllMode_)
```

Note the `LINK` and `DLL` attributes. These are the little critters that cause so much grief. They're there because a DLL that exports a class has to handle that class differently than a DLL or EXE that uses the exported class.

The help has this to say about LINK:

LINK	Names a file to add to the link list for the current project.
Linkfil	A string constant naming an file (without an extension .OBJ is assumed) to link into the project. Normally, this would be the same as the parameter to the MODULE attribute (which by default is a source file), but may explicitly name a .LIB or .OBJ file.
Flag	A numeric constant, equate, or Project system define which specifies the attribute as active or not. If the flag is zero or omitted, the attribute is not active, just as if it were not present. If the flag is any value other than zero, the attribute is active.

Let's say you're creating a hand coded DLL that will contain all your generic classes (easy to do, I assure you - just wait a bit). In that case you'll want the `LINK` attribute to be active, because you want the class source to be compiled and linked.

If you're using that class in another app, one that will make use of the DLL with the class source compiled in, you want `LINK` to be inactive, because your app is going to make calls directly into the generic class DLL. You don't want to link in the source.

This is why you'll see `_ABCLinkMode_` defined as `true` (1) in an app that exports the ABC classes (typically also a data DLL), and defined as 0 in an app that makes use of that base class/data DLL.

But it's not enough to take care of the linking differences. There's also the `DLL` flag, about which the help says this:

DLL	Declares a variable, FILE, QUEUE, GROUP, or CLASS defined externally in a .DLL.
flag	A numeric constant, equate, or Project system define which specifies the attribute as active or not. If the flag is zero, the attribute is not active, just as if it were not present. If the flag is any value other than zero, the attribute is active.

The help also notes that "The DLL attribute is required for 32-bit applications because .DLLs are relocatable in a 32-bit flat address space, which requires one extra dereference by the compiler to address the variable."

This translates loosely as "If you're using a class exported from a DLL, and you don't have an active DLL attribute on the class, your app is gonna blow chunks."

The problem I have is I can never seem to keep `_ABCLinkMode_` and `_ABCDllMode_` straight in my head. Objectively I know that a DLL that exports the ABC classes needs the defines

```
_ABCDllMode_=>0;_ABCLinkMode_=>1
```

while a DLL or EXE that uses that DLL with the exports needs the defines

```
_ABCDllMode_=>1;_ABCLinkMode_=>0
```

but it doesn't matter. It's only two variables, each with just two states, but my eyes glaze over anyway.

What complicates this further is that in my own code I don't want to use `_ABCLinkMode_` and `_ABCDllMode_`, because my classes are potentially useful in different applications (not just APP files). If I have two different applications, each made up of a data DLL, non-data DLLs, and one or more EXES, and I add a new class or method to my class library (which happens all the time), I now have to update the exports for each of those data DLLs. I don't want to do this - I want a single point of maintenance for my classes.

Third party vendors have the same problem - they don't want you to have to export their classes from your data DLL (the one that typically exports ABC classes). So they set up their own equates, which have to have their own project defines to ensure that their classes are handled correctly. I've seen apps with numerous pairs of DLL and LINK mode defines as needed by the various third party products.

This, not to put too fine a point on it, is nuts. In most circumstances there's absolutely no need for *any* project defines in any app *except* the one that compiles and exports the class(es).

A better way

I've only touched on two uses of the `LINK` and `DLL` attributes - linking classes into a DLL, and using classes exported from such a DLL. There are other possibilities such as linking in classes in an OBJ or LIB, but these aren't very common. I strongly suspect that almost everyone who reuses classes does so by exporting from one DLL and then using those exported classes in other DLLs and EXEs. This is the scenario I'm addressing.

Here's how you create classes that don't need any special treatment to be used in any way, with the exception of a single project define in the project that creates the class DLL.

Step 1: Create a hand coded DLL

Choose File|New Solution, Project or Application. Provide a name and select the Win32 DLL quick

start.

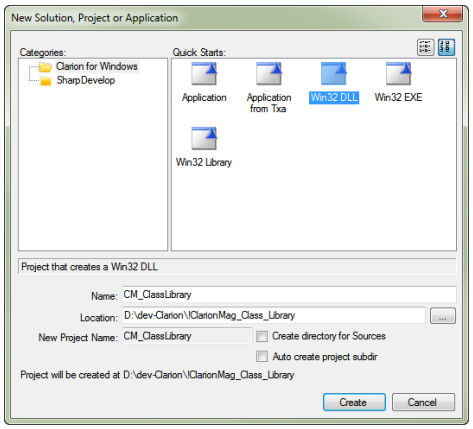


Figure 1. Creating a classes DLL

Figure 2 shows the resulting source files. There are just two: the program CLW file and the EXP file which will contain the definitions of the things you want to export.

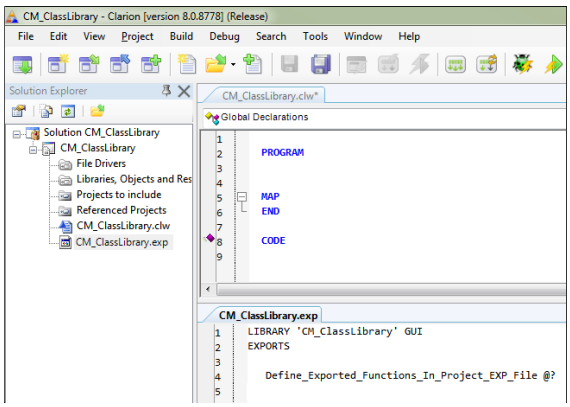


Figure 2. The program and EXP files

Step 2: Create a class

For purposes of this article, my class library will export `CM_System_String`, which is an updated version of Rick Martin's string class.

The class header looks like this:

```
CM_System_String    CLASS,TYPE,MODULE('CM_System_String.CLW'),|
                      LINK('CM_System_String.CLW',_CM_Classes_LinkMode_),DLL(_CM_Classes_DllMc
```

As you can see I've created my own equates for the `LINK` and `DLL` attributes. But I'm not going to define these in any projects anywhere. Instead, at the very top of the CM_System_String.inc file I have this line of code:

```
  include('CM_IncludeInAllClassHeaderFiles.inc'),once
```

The file contains the following:

```
   OMIT('***',_Export_ClarionMag_Classes_)
_CM_Classes_LinkMode_                 equate(0)
_CM_Classes_DllMode_                  equate(1)
    ***

    COMPILE('***',_Export_ClarionMag_Classes_)
_CM_Classes_LinkMode_                 equate(1)
_CM_Classes_DllMode_                  equate(0)
```

* * *

I've made the problematic `LINK` and `DLL` mode equates subject to a single, more readable `_Export_ClarionMag_Classes_` equate. If this equate is absent or set to 0, then the classes are not compiled, and are defined as being in another DLL. If the equate is present and set to 1 then the classes are compiled into that DLL.

Consequently, there is only one place I ever need to set a project define, and that's in the DLL that exports the classes.

Step 3: Set the project define

From the solution explorer, right-click on the project and choose Properties (Figure 3).

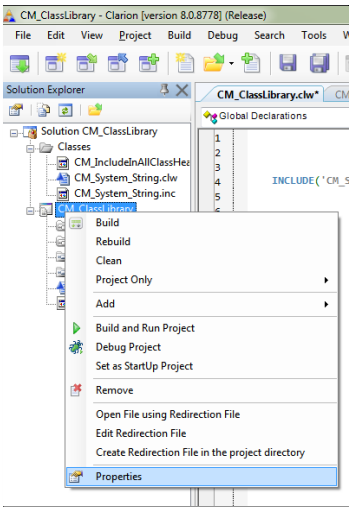


Figure 3. Opening the project properties

On the Compiling tab, for Conditional Compilation Symbols (a.k.a. project defines), I add the text `_Export_ClarionMag_Classes_=>1`.

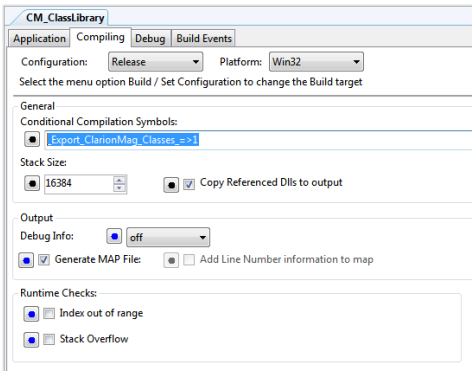


Figure 4. Setting _Export_ClarionMag_Classes_=>1

This is the *only* place I set the `_Export_ClarionMag_Classes_` value. I *never* need to do it in any app where I use my class library DLL.

Step 4. Update the source and EXP and compile

There isn't much to creating the DLL itself. In the program source file I add an include statement pointing to my class header:

```
                    PROGRAM


    INCLUDE('CM_System_String.INC'),ONCE
```

```
                                        MAP
                                        END

    CODE
```

And I need to add the exports to the EXP file. I created these by copying and modifying the linker errors.

```
LIBRARY 'CM_ClassLibrary' GUI
EXPORTS

    APPEND@F16CM_SYSTEM_STRINGsb  @?
    APPENDLINE@F16CM_SYSTEM_STRING  @?
    APPENDLINE@F16CM_SYSTEM_STRINGsb  @?
    ASSIGN@F16CM_SYSTEM_STRINGsb  @?
    ASSIGNTOLINE@F16CM_SYSTEM_STRINGsbl @?
    BEGINSWITH@F16CM_SYSTEM_STRINGsb  @?
    DESTRUCT@F16CM_SYSTEM_STRING  @?
    ENDSWITH@F16CM_SYSTEM_STRINGsb  @?
    GET@F16CM_SYSTEM_STRING  @?
    GETALLLINES@F16CM_SYSTEM_STRINGOsb @?
    GETLINE@F16CM_SYSTEM_STRINGl  @?
    INDEXOF@F16CM_SYSTEM_STRINGsbl  @?
    INSERTAT@F16CM_SYSTEM_STRINGsbl  @?
    LASTINDEXOF@F16CM_SYSTEM_STRINGsb  @?
    PREPEND@F16CM_SYSTEM_STRINGsb  @?
    RECORDS@F16CM_SYSTEM_STRING  @?
    REPLACE@F16CM_SYSTEM_STRINGsbsbOlOl  @?
    REPLACEWORD@F16CM_SYSTEM_STRINGsbsbOl @?
    SPLIT@F16CM_SYSTEM_STRINGBq  @?
    SPLIT@F16CM_SYSTEM_STRINGsb  @?
    SUBSTRING@F16CM_SYSTEM_STRINGll  @?
    TRIM@F16CM_SYSTEM_STRING  @?
    TYPE$CM_SYSTEM_STRING  @?
    VMT$CM_SYSTEM_STRING  @?
```

All I need to do is compile the solution, and I've created a DLL with my exported class, one I can use in any DLL or EXE without being concerned about setting project defines.

Extra stuff

I also have a batch file set as a post-build task. The contents of that file are as follows:

```
copy CM_ClassLibrary.dll ..\bin
copy CM_ClassLibrary.dll ..\unittests
copy obj\release\CM_ClassLibrary.lib ..\lib
```

I copy the DLL and LIB files to standard locations (which are specified in my redirection file). And because I use ClarionTest for unit testing, I copy the DLL to the directory where I keep the ClarionTest executable, since any tests that reference the class library will look for the DLL in that directory.

I also use John Hickey's excellent ClarionLive! Class Creator (a.k.a. Clive's Class Creator) to quickly and easily create new classes from my own templates. My standard class INC file looks like this:

```
    include('CM_IncludeInAllClassHeaderFiles.inc'),Once

CM_BaseClass       Class,Type,Module('CM_BaseClass.CLW'),|
                        Link('CM_BaseClass.CLW',_CM_Classes_LinkMode_),Dll(_CM_Classes_DllMode
Construct             Procedure()
Destruct              Procedure()
                   End
```

Clive creates my new class using this class as a template (not as a literal base class). As you can see I have my include in place as well as the appropriate `LINK` and `DLL` attributes.

Summary

For years I've found the `DLL` and `LINK` attributes to be a huge pain, and on more than one occasion I've either forgotten to add them or added them incorrectly, with predictably ugly results.

The technique I've described in this article eliminates the need to set project defines to use exported classes. It takes a little more setup (which I've now completely automated using Clive), but only requires a single compilation symbol and that only for the DLL that exports the class. I can use my exported classes in any app without being concerned at all about whether I have the correct project defines set up.

Download the source

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

Article comments

BACK TO TOP

Populating Lists From Queues In Clarion 7/8

By Steven Parker

Posted November 28 2011

In all versions of Clarion for Windows, up to Clarion 7, creating a list box from a queue was easy. A bit labor intensive but easy.

In C6, for example, on the top menu, select Control (not Populate) then List Box.

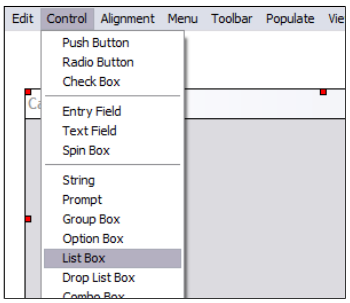


Figure 1. Select Control

On the Select Control Template dialog, which appears when the cross hairs to place the control appear, tick “Populate control without control template:”

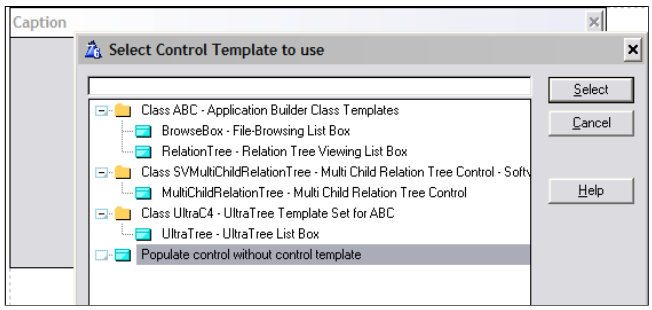


Figure 2. Browse Box without a template

Cancel out of the “Select Column” dialog and, on the Property List dialog, which appears next, right click list box and name a queue in “From:”

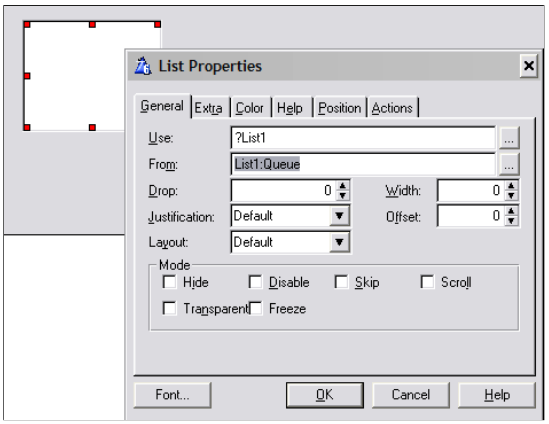


Figure 3. Nominating your own queue

And, voilà, the list box will be filled from my queue.

Enter Clarion 7

Everything changed with Clarion 7. The community had been clamoring for a more Visual Studio-like IDE in addition to 32 bit-ness. Softvelocity listened and delivered a VS-like IDE.

Well, the VS-esque IDE eliminated the “Control” menu and the “Populate control without a template” prompt (and option). But, not only is the ability to populate a control as we did from the “Control” menu still available, making list box based on a queue is actually easier than it was in earlier versions of Clarion.

I know a list can display a queue because I did it in a previous article (see “Queue Record Structures” – in fact, the sample app, downloadable at the end of this article, a bit updated, serves as the demo for this article too). So, let's see how to browse queues in Clarion 8 (8.0.8658, specifically).

The “Control” Menu

As documented in “Queues in Dictionaries,” start with a Window procedure (from the Templates tab, not from the Defaults tab that I would normally use when creating a new procedure).

I want to populate a browse box. But I do not want to use the BrowseBox from the Control Templates pad (in Clarion 6, this would be equivalent to Populate | Control Template and selecting “Browse Box”). The Control Templates BrowseBox assumes I will use a file, not a queue (though this can be made to work for browsing a queue, the work involved is counterproductive).

In Clarion 8, use the Toolbox:

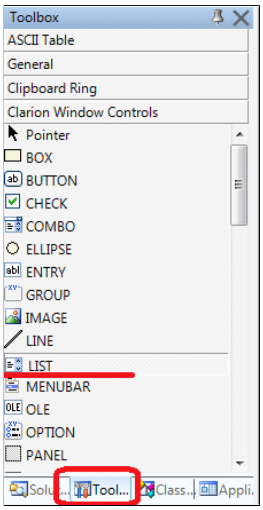


Figure 4. The Toolbox

This is the equivalent of Clarion 6's Control | Listbox menu.

"Populate control without control template"

If I drag a List from the Toolbox to the window and drop it there, I do not get the "Populate control without control template" prompt. And this is very confusing.

It turns out, I don't need it. I can do everything I need to do from the list's Properties pad.

As expected, a USE variable is created for the list control, of course. The FROM attribute is also available on this pad. All I need to do is change the FROM:

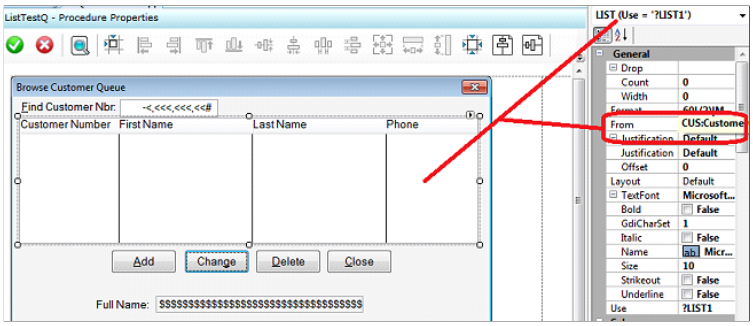


Figure 5. List Properties

(In this illustration "CUS:CustomerQ" is the LABEL of a queue created in the dictionary. But a queue declared in any of the data pads or a data embed will do.)

There is an ellipsis that I can use to select a queue LABEL from a data pad (global, module or local). Or, I can, assuming I remember, simply type in the LABEL of my queue (if the queue is declared in a data embed, this is a necessity).

If the queue is declared in the dictionary, I can populate the list box just like any file-based browse box. If the queue is declared in a data embed or if I do not populate the box, all the queue elements will display, just as in Clarion 6 and earlier.

Summary

We wanted a Visual Studio look and feel to our 32 bit IDE. We got it. Almost everything the IDE used to feature is still there. But many things, many, have moved to new locations. You just have to look for them.

Populating a list from a queue, however, is actually easier than before.

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

BACK TO TOP

Thread-Safe Runtime Translation With The In-Memory Driver

By Nardus Swanevelder

Posted November 30 2011

Towards the end of 2003 I wrote a series of articles on Clarion's Translation Class. I discussed what you need to do to implement translation functionality into your application and I showed you how to add code to your application that will enable you to change your search and replace strings at runtime.

What is this search and replace strings about? If your application is used by various clients in the same industry but your clients have different descriptions for the same concept, what do you do? Clarion's Translation Class works on the basis that you supply it with a search string and the RTL will replace that with the replace string. For example one of my clients refers to a child company as a "Line Of Business" but another client is referring to it as a "Centre Of Excellence". Now I can create two identical applications and change all of the "Line Of Business" strings to "Centre Of Excellence", but that is a lot of work and worst of all it creates lots of duplication. So Clarion's Translation Class to the rescue.

My only problem with Clarion's Translation Class is that I can't change the translation strings at runtime. But that short coming I addressed by creating a table or two and adding a template to the application.

One of the problems with my original approach was the fact that I used a Global queue to store the search and replace strings; as you are hopefully aware by now, global queues can be a problem in Clarion 6/7/8 with the new threading model.

So what to do, what to do? One option is to encapsulate global data assignments in a critical section.

The second option is to use SoftVelocity's In-Memory Database Driver (IMDD), which is the path I'll explore in this article. The IMDD is thread safe due to the fact that it is a file driver, and the File Manager class is thread safe. The only draw back with the IMDD driver is that you have to purchase it from SoftVelocity. It does not ship as part of Clarion Personal Edition or Clarion Enterprise Edition. The IMDD can be used for much more than making your Global queues thread safe but that is a discussion that does not form part of this article.

I am using the IMDD driver in most of my applications and recently I decided to upgrade my Translation template to use the IMDD driver where I had previously used a queue. I have left the option to use global queues in the template for backward compatibility, but please take note that if you use the global queues in this template that it is not thread safe.

The basics of translation

For those of you who did not read the previous series of articles, and for those of you who can't remember them anymore, here's a quick recap.

I use two tables to store my translation strings externally to my application. The first table stores the Language Settings; Figures 1 and 2 show the Browse and Update screens for this table.

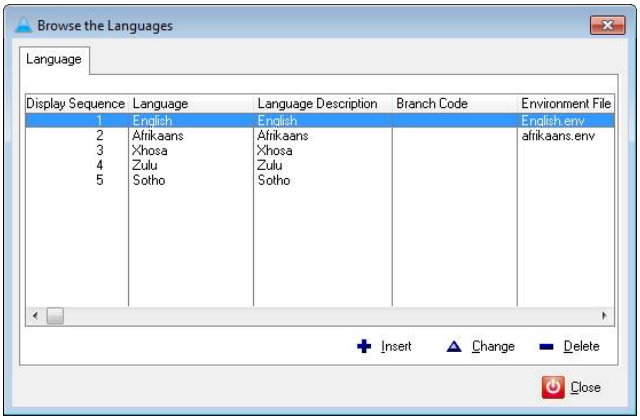


Figure 1. The Browse screen for the Language File

Figure 2. Update Screen for Language File

The second table stores the Search and Replace Strings.

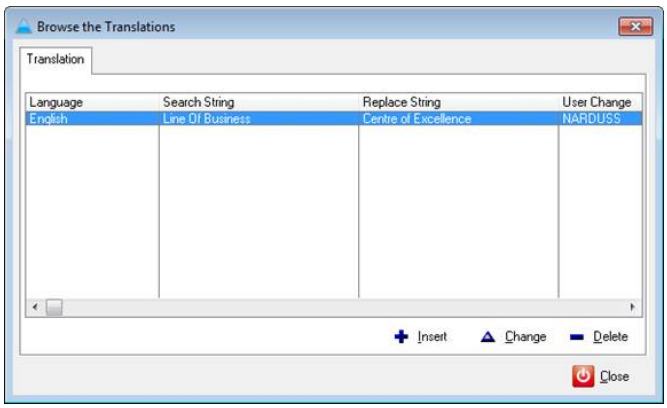


Figure 3. The Browse screen for the Translation File

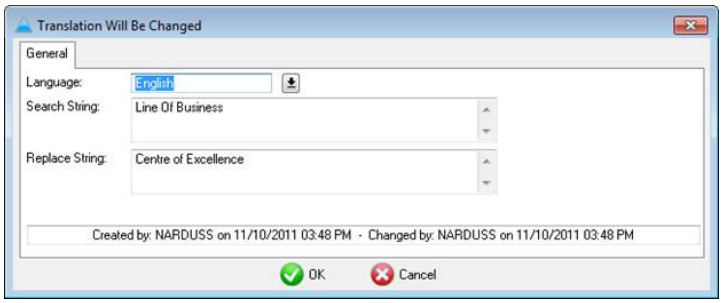


Figure 4. The Update screen for the Translation File

Adding the template to your Application

Now you know how to populate the application with Translation strings. The next step is to look at what is involved in adding this functionality to your application.

Step 1 – Add tables to your dictionary

You have to add the two tables as discussed above, and if you are going to use the IMDD, you also have to import the one IMDD table that corresponds to the queue in the non-IMDD version.

Step 2 – Global Template

Add DinamiComp's Translation global template (included in the downloadable source, and free for your use) to your application's global extensions. Complete the Database Tables Tab as well as the Queue/In-Memory tab. In a multi-DLL application you need to add the global extension to each of your apps where you need the translation functionality.

Figure 5. Main Global Template

Click on the Database Tables Tab to start configuring the template.

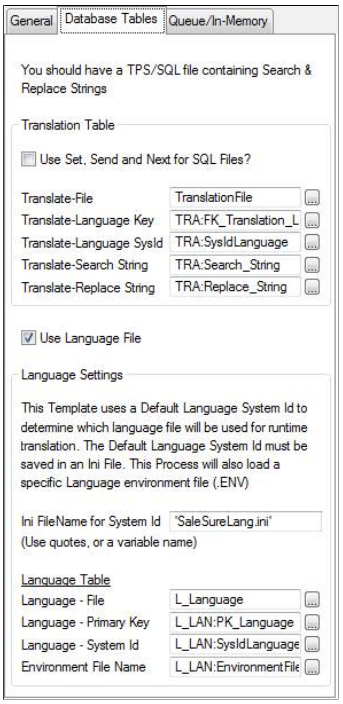


Figure 6. Database Tab

The Language file is mainly used to determine which Language must be used for the Translation at runtime.

In a MS SQL environment you also have the option to use the Set, Send and Next File commands rather than the Access:FileName commands.

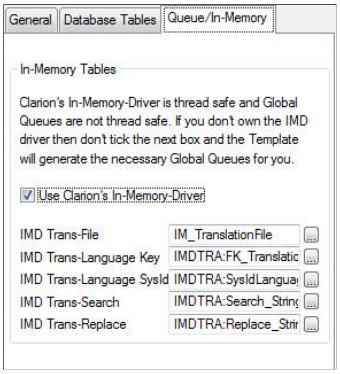


Figure 7. Population of In-Memory Data Driver Info on Queue/In-Memory Tab

If you are going to use an In-memory file, you have to specify the In-memory file details that correspond with the Translation File. If you choose not to make use of the In-memory file the template will create a queue to keep the search and replace strings.

Step 3 – Add Translation Browse and Update screens to your application

This step is only required if you believe that the client has sufficient skills/knowledge to have access to this function. See figures 3 and 4 for examples of these screens

Step 4 – Enable Clarion's Translation

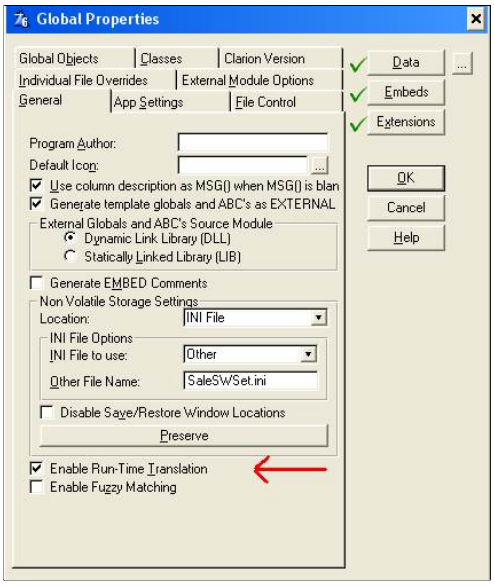


Figure 8. Enabling Runtime Translation in Clarion 6

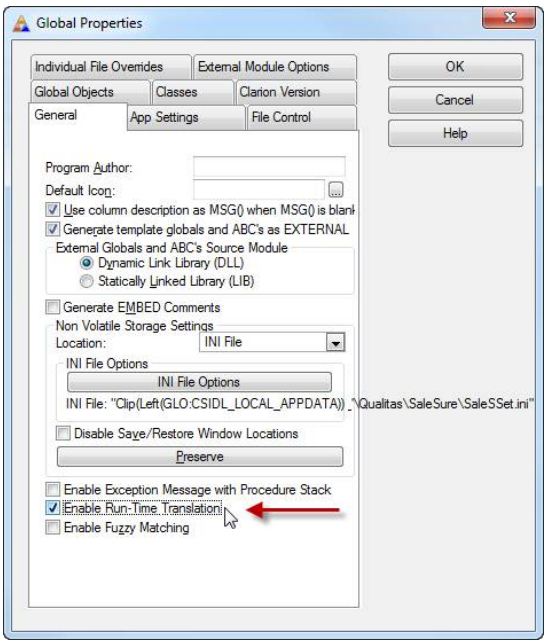


Figure 9. Enabling Runtime Translation in Clarion 8

Step 5 – Compile your Application

If you are using the In-Memory Driver you have to tick the ‘Generate file declaration’ for the In-memory file under the Individual File Overrides Global Properties. (You only have to do this in the EXE application)

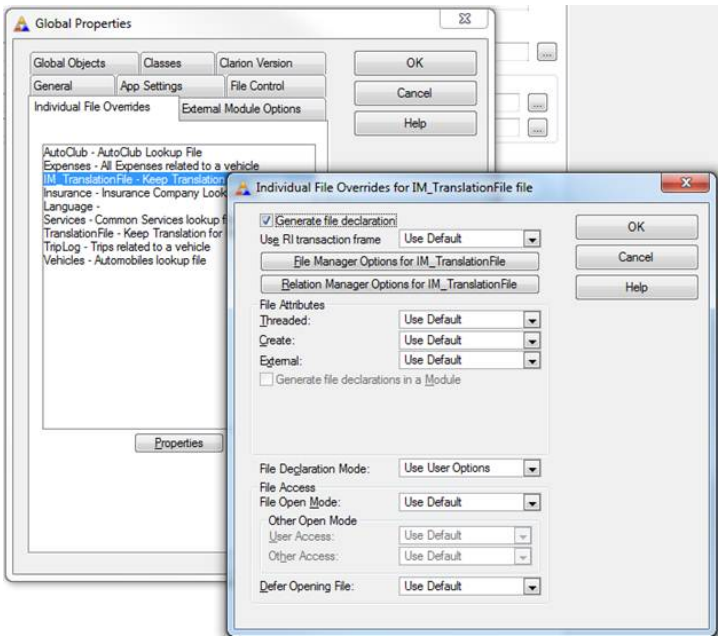


Figure 10. File overrides

No need to do anything else; the template takes care of the rest.

The Template adds code to the following areas:

`AT(GlobalDATA)`

- Defines global variables and queue (If using queue)

`#AT(%BeginningExports)`

- In a Multi-dll app the template exports the necessary symbols

`#AT(%ProgramSetup)` - data dll

- Opens Ini file to get Language setting
- Opens and gets the language file
- Sets the Locale()
- Opens translation and IMDD files
- Copies translations from translation file to IMDD file
- Adds translation strings to Clarion's Translator

`#AT(%ProgramSetup)` - other dlls and Exe

- Opens IMDD file
- Add translation strings to Clarion's Translator

Summary

In my previous series on Clarion Translation I showed how you can extend Clarion's Translation functionality to use search and replace strings defined in tables. In this way you can supply new translations to your customers without having to update the application itself – instead, you simply send along an updated table with a set of search and replace strings. In this article I added a template to

enable you to use the In-Memory Database Driver in stead of Global Queues, for a thread-safe solution.

Download the source

Nardus Swanevelder was born and raised in South Africa. He was a networking engineer for seven years before he moved over to the commercial side of the business. Nardus has developed a Sale Cycle Management system for the Information and Communication Technology industry. He has been programming in Clarion since 1989, and holds B.Com and MBA degrees. In his spare time Nardus lectures Financial Management to B. Com Hons students at North-West University.

Article comments

BACK TO TOP

Passing Queues

By Steven Parker

Posted November 30 2011

Having recently conquered – perhaps "subdued" or "quelled" might be better descriptions – passing GROUPs not only between procedures but between threads, there remains one more complex structure. My new Everest is Queues (without concern as to passing them across threads).

It's not simply that I feel somehow compelled to finish a perceived "set" of Clarion skills. I have a real need, I think. It's a long story and I won't bore you with it. So let's get to passing queues around an app.

The Plan

What I need to do is to read a set of records from a file. I need to do several kinds of subtotaling based on fields in the record. I probably need to reprocess the queue to get additional totals after the initial loading of the queue (note that because I want to post-process within the structure, the IMDD, which I would ordinarily use, is not entirely appropriate; besides, my target app is in 9033 and I can't find an (IMDD install for 9033).

Queues are great for this sort of thing. But because "globals are bad," I want to use a Named Queue and pass a single (local) queue around an app. I hijacked the data files from the People.APP (People.TPS). It comes with a number of records and has a field (Gender) for testing subtotaling.

I need:

- a Process template procedure to read People.TPS and load my queue
- a browse type procedure to view the queue contents
- a Report Template procedure to ensure that I can make reports on my queue

I also want to test whether I can use the List Box Formatter with a queue parameter.

My test app, then, will start with the queue browser. Early on, I will call the Process to build the queue.

I will, if possible, use the List Box Formatter in the browse procedure. So, I'll also do a non-formatted list (where the List's FROM attribute is the Label of a queue and the Formatter is not invoked at all) just to see if there is a difference between the two browses. This second browse will be called from a button on the "main" browse.

Similarly, I will make a report from the queue. The report, too, will be called from a button on the "main" browse.

Queue Parameters

The demonstration application, which you can download at the end of this article, is made in 8.0.8740;

all run times are provided for those who do not yet have C8.

As with GROUP parameters, the first step is to create the new data type. That is, I need to create the TYPEd queue declaration. I chose to do this in the global Data Pad. While a global data embed is perfectly adequate, using the Data Pad does write parts of the declaration for me, avoiding typotunities. Using the Data Pad also makes my data available for Window and Report Formatters, should I want to populate a window or report.

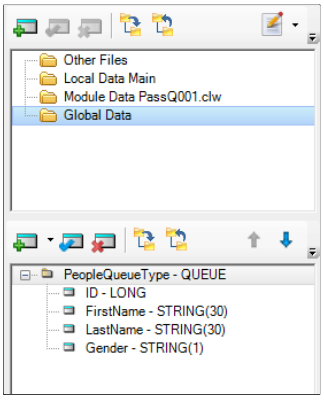


Figure 1: Global Declaration

The next step is to create the target browse. Because this is not file browser, I use the Window template with a List Control populated on it.

Before I can populate a List Control from Control Pad, of course, I have to make a local queue derived from the TYPEd queue. Either the Local Data Pad or a data embed can be used. Again, I choose to use the local data pad because I may want to use the data in the Window Formatter:

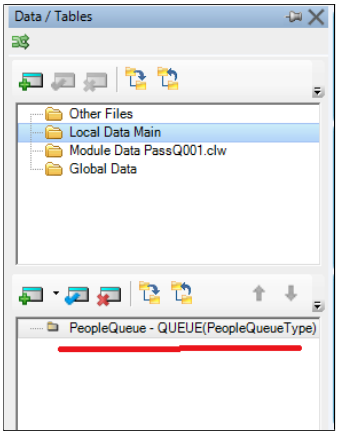


Figure 2: Locally derived "copy" of queue

Jumping ahead a bit, I didn't think about the fact that because I use the "Base Type" option in the Data Pad, the individual elements of the queue would not appear in the Local Data Pad. Because of that, I can't use the local queue in the List Box Formatter after all. But I could and did use the global declaration successfully in the List Box Formatter.

Before the List Box displays, I need to load People.TPS into my queue. In INIT, I call a Process template procedure to do this. I pass the local queue:

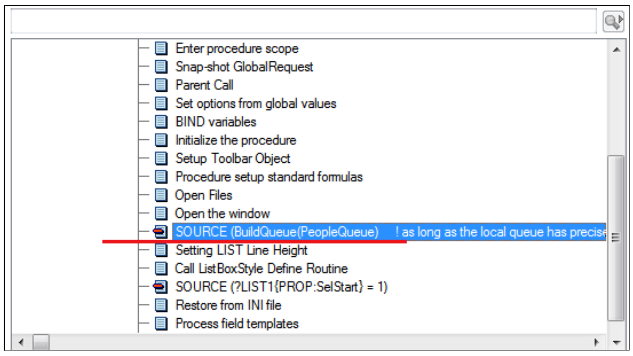


Figure 3: Call Process which creates the queue

Inside the Process procedure, I use dot syntax to populate the queue:

```
pPeopleQueue.ID = PEO:Id
pPeopleQueue.Firstname = PEO:FirstName
pPeopleQueue.LastName = PEO:LastName
pPeopleQueue.Gender = PEO:Gender
ADD(pPeopleQueue)
```

and to do some post-processing in the `KILL` method:

```
Case PEO:Gender
  Of 'M'
    pPeopleQueue.FirstName = '   Total Males'
    Get(pPeopleQueue,pPeopleQueue.FirstName)
    If ErrorCode()
      pPeopleQueue.ID = Max + 10
      pPeopleQueue.LastName = 1
      pPeopleQueue.Gender = ''
      Add(pPeopleQueue)
    ELSE
      pPeopleQueue.LastName += 1
      Put(pPeopleQueue)
    End
  Of 'F'
     ! more code here
  End
```

All of this is pretty standard stuff.

`BuildQueue` is prototyped with both Labels, just like `GROUP` parameters.

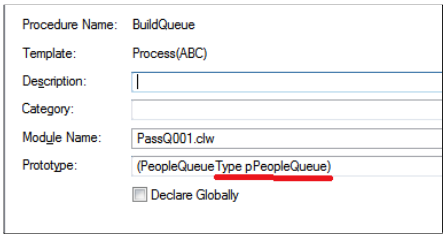


Figure 4: Prototyping the Process

Back in the List Box, do not use the `IMM` attribute on the List control. With the Immediate attribute set, the List will not respond to navigation keys:

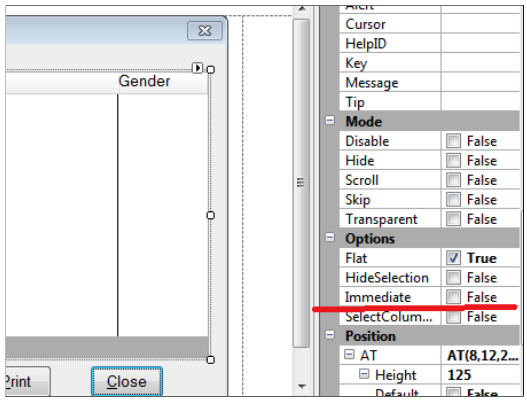


Figure 5: Turn off Immediate attribute on List

If this works, opening PassQ.EXE should bring up the progress window for the Process and then you should see a list box filled with the names from People.TPS (and the additional subtotals I made). Try it.

Reporting from a Passed Queue

The report is called using the standard Actions for a button. I pass the local queue Label to the report:

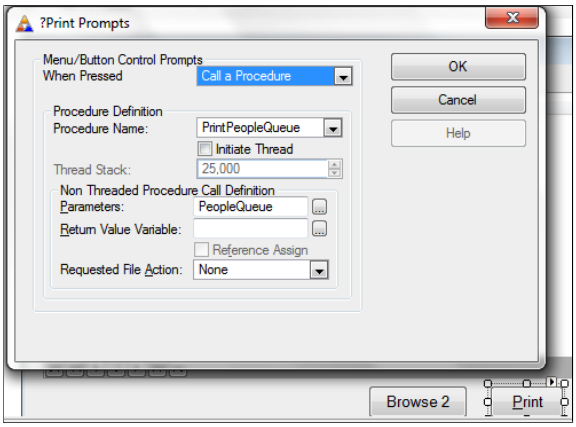


Figure 6: Button Actions to call the Report

The Report is prototyped in the same way as the Process.

Populating the report fields is where things get interesting … and instructive. In previous versions of Clarion, populating a String Control onto a window or report and ticking the "Variable" check box was easy, **if** the variable is visible to the Formatters (hence, my predilection for using the Data Pads). But I always found it difficult to populate variables declared in embeds and, therefore, not visible to the various Formatters.

Clarion 8 makes this all a bit easier, provided I remember the Labels of my variables.

From the Toolbox Pad, move a String Control onto the report. With the string selected, there are two entries on the Properties Pad that make this "do-able."

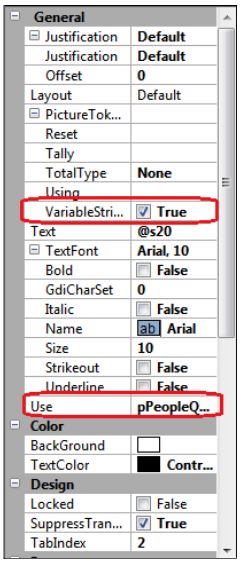


Figure 7: String Control's Properties Pad

Well hidden (to these old eyes), under General | Layout is "VariableString," a check box. Tick it.

Just below, in the Use prompt, I type in my variable's Label and the "screen picture" ("Text" property) is filled in correctly. This is much easier than in the old days.

Summary

It is quite a bit less work to declare a global queue, I suppose. But that also bloats global memory requirements.

Queue parameters, luckily, are not substantially different that `GROUP` parameters. Queues, of course, can be used in Lists, where `GROUP`s cannot, and not having a bunch of them floating around an app (`THREAD`ing the `TYPE` declaration works quite nicely) really does make for a more elegant – and flexible – app.

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

BACK TO TOP

Tip of the Week: Defeating Circular Dependencies

By David Harms

Posted November 30 2011

If you're one of those unfortunate souls dealing with apps containing circular references, and you haven't yet worked up the nerve to untangle your code, you may have noticed that you can sometimes have problems compiling apps with circular dependencies when you use the Solution Explorer.

If you right-click on a project node in the Solution Explorer and choose Build, and that project contains circular dependencies, you'll get an error message and the project will not build.

When I first encountered this I was more than a little annoyed, because I sometimes like to make modifications to generated source when debugging. That way I can hack and slash at my code, and once I find the error (or if I don't) I can simply make the needed correction to the APP and regenerate the source. In a few situations, the circular references prevented me from using this technique.

There are two ways around the problem. Since MS Build (the underlying build system) detects circular dependencies via project includes, you can change any included projects to simple included LIB files. But that can be a lot of work, and your changes may be overwritten on the next generation.

A simple approach is to use the App Pad. You don't have to generate apps, you can choose to just compile.

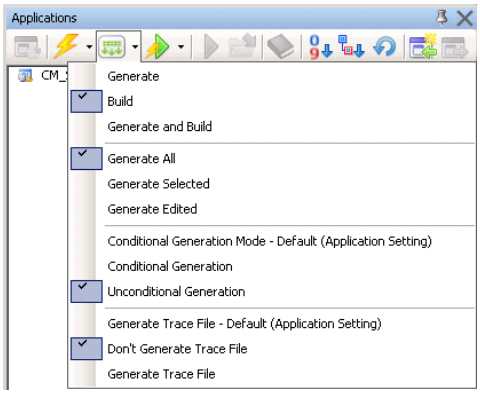


Figure 1. Setting the App Pad to Build only

Note that Build is selected, rather than Generate and Build. This is all that matters - if that second option is checked, there will be no generation.

For whatever reason, compiling apps with circular dependencies from the App Pad doesn't get MS Build's nose. That's not as good as fixing all the circular references, of course, but sometimes you really

do have to settle for second best.

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

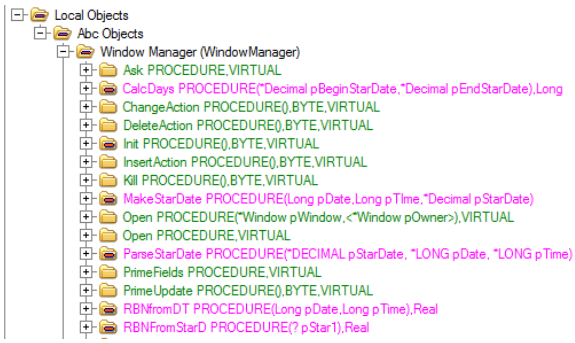
BACK TO TOP

Custom Methods And Derived Classes

By Nardus Swanevelder

Posted November 30 2011

I recently read an article by Steve Parker about dates and doing calculations with dates. When I downloaded the source for the article I saw the following embed tree in the sample application:



Looking at the above embed tree raised some questions in my mind. Questions like: what are these pink procedures, why do I need them and lastly how do I add them?

What are these pink procedures?

The pink procedures are Steve's own methods that he added into the derived `WindowManager` class.

Why do I need them?

Steve found that doing it this way helps him to prototype the procedures so that he can just copy and paste the declarations into his own class.

If you go into the source of the `CompareDates` procedure you will see the following code:

```
ThisWindow          CLASS(WindowManager)
Ask                   PROCEDURE(),DERIVED
ChangeAction          PROCEDURE(),BYTE,DERIVED
Open                  PROCEDURE(),DERIVED
PrimeUpdate           PROCEDURE(),BYTE,PROC,DERIVED
Reset                 PROCEDURE(BYTE Force=0),DERIVED
Run                   PROCEDURE(),BYTE,PROC,DERIVED
SetAlerts             PROCEDURE(),DERIVED
TakeAccepted          PROCEDURE(),BYTE,PROC,DERIVED
TakeDisableButton     PROCEDURE(SIGNED Control,BYTE MakeDisable),DERIVED
TakeEvent             PROCEDURE(),BYTE,PROC,DERIVED
Update                PROCEDURE(),DERIVED
RBNfromDT             PROCEDURE(Long pDate,Long pTime),Real ! New method added to this class instance
RBNFromStarD          PROCEDURE(? pStar1),Real              ! New method added to this class instance
ParseStarDate         PROCEDURE(*DECIMAL pStarDate, *LONG pDate, *LONG pTime)  ! New method added to this class instance
MakeStarDate          PROCEDURE(Long pDate,Long pTime,*Decimal pStarDate) ! New method added to this class instance
TimeSplit             PROCEDURE(*Decimal pTime,*Long pDays,*Long pHours,*Long pMinutes,*Long pSeconds) ! New method added to this class instance
CalcDays              PROCEDURE(*Decimal pBeginStarDate,*Decimal pEndStarDate),Long ! New method added to this class instance
                    END
```

If you compare the `MakeStarDate`, `ParseStarDate`, `TimeSplit` and `CalcDays` procedures above with Steve's shpStarClass.Inc below, you will see that they are exactly the same.

```
shpStarClass    CLASS(shpTimeClass),TYPE,MODULE('shpStarClass.clw'),LINK('shpStarClass.clw',1)
Construct               PROCEDURE
Destruct                PROCEDURE
MakeStarDate            PROCEDURE(Long pDate,Long pTIme,*Decimal pStarDate)
StarET                  PROCEDURE(*Decimal pStarDateBegin,*Decimal pStarDateEnd),String
TimeUpTime              Procedure(Real pTimeAllowed),Real
ParseStarDate           PROCEDURE(*DECIMAL pStarDate,*LONG pDate,*LONG pTime)
ParseStarDate           PROCEDURE(Real pStarDate,*LONG pDate,*LONG pTime)    ! override, needs Real if called after StarET
CalcDays                 Procedure(*Decimal pBeginStarDate,*Decimal pEndStarDate),Long
                End
```

Steve found one reason for adding your own methods into the `WindowManager` class, but the more common reason is so you can extend the functionality of `WindowManager` by adding your own methods (procedures).

Besides creating new functions you can also create overloaded functions – these are procedures with the same label as existing procedures but with a different prototype

How do I add new methods?

Adding your own procedures or methods into the `WindowManager` is fairly easy. Just follow these steps:

Click on the Properties Tab of the Procedure where you want to add the new procedure(s).

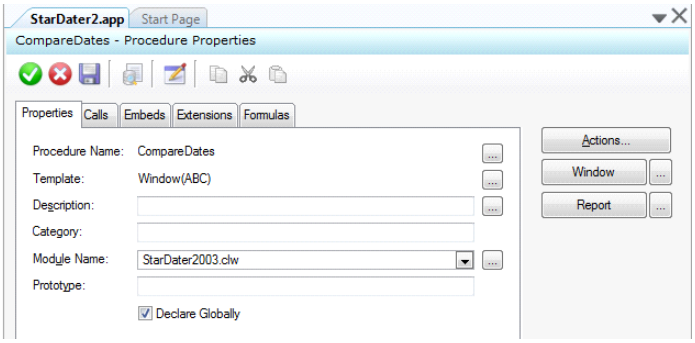


Figure 1. CompareDates Procedure

Clicking on the Actions button will display the properties screen for the Procedure.

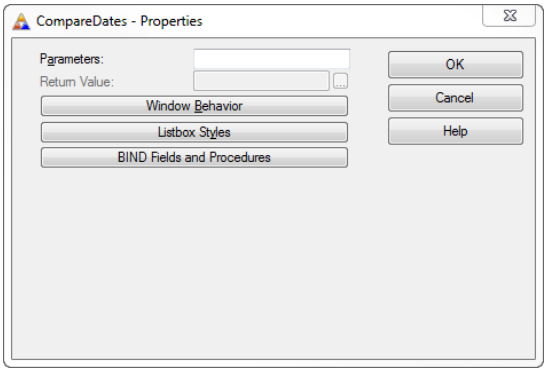


Figure 2. Properties screen for a Procedure

Click on the Window Behaviour button, and then click on the Classes Tab. You will see two buttons: New Class Methods and New Class Properties. These two buttons are disabled by default and to enable them you have to tick the Derived tick box. When you click the derive tick box it basically tells Clarion that you are deriving a new class from the `WindowManager` (actually in most cases the derived class already exists) and that you will be adding new methods and/or properties to the class.

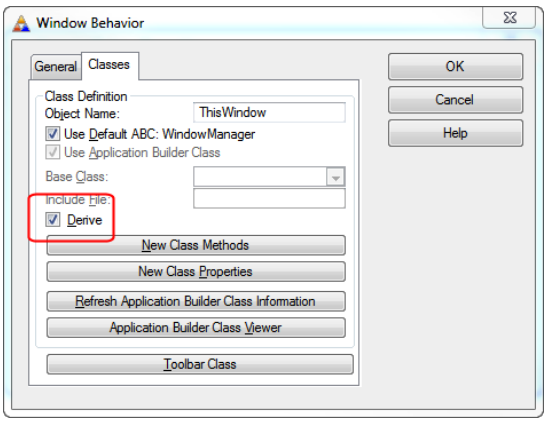


Figure 3. Window Behaviour screen

When you click on the New Class Methods button you will be presented with a screen like this:

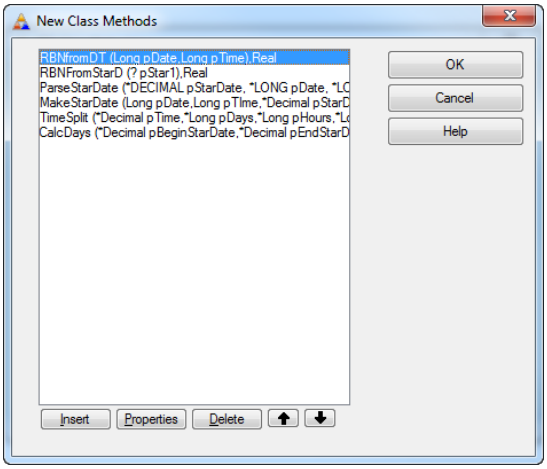


Figure 4. New Class Methods

When you select one of the above methods and you click on the Properties button you will see this screen:

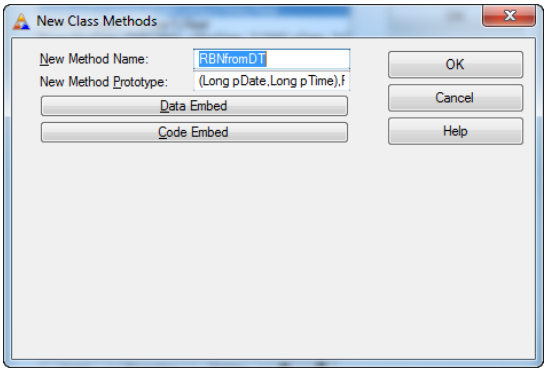


Figure 5. Defining a new Class Method

To add properties to this procedure, click on the Data Embed button:

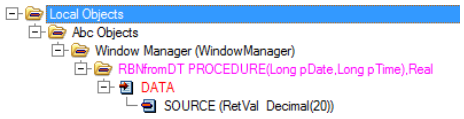


Figure 6. The data embed

Double clicking on the Source line gives you the properties defined for this method:

```
1  RetVal  Decimal(20)
2  D1      Decimal(20)
3  T1      Decimal(20)
4
```

Figure 7. The generated properties

You can add new properties here.

To add the code for the method click on the Code Embed button:

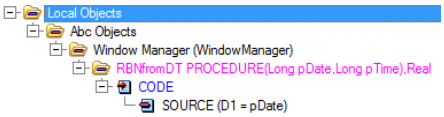


Figure 8. The code embed

And double clicking on the Source line gives you the source code for this method:

```
1  D1 = pDate
2  T1 = pTime
3  Return T1 + (D1 * eDay)
4
```

Figure 9. The method source

If you want to add properties that are “global” to the class click on the New Class Properties Button:

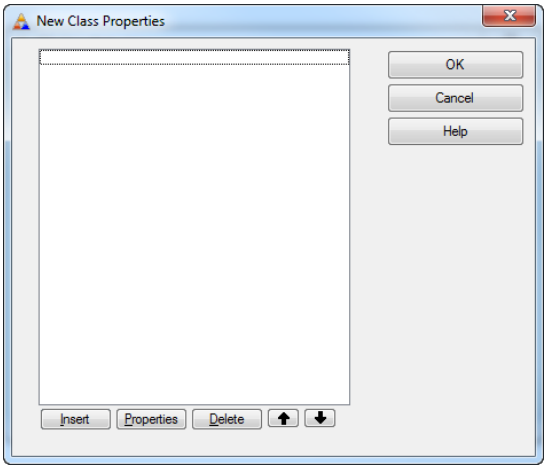


Figure 10. New Class Properties

Click on the Insert button to define a property for this class.

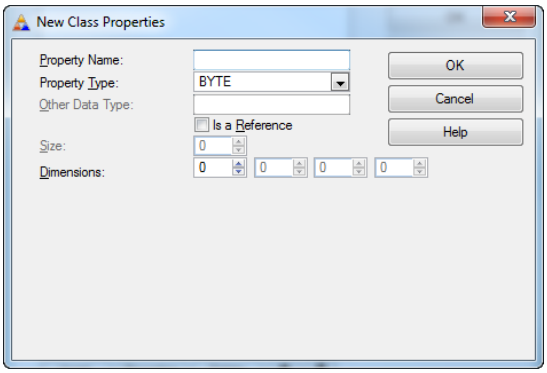


Figure 11. Defining a new Class Property

An example

Let us look at a simple example.

In the attached example open the school.app file, go to the `BrowseStudents` procedure and open it.

Click on the Actions Button, Click on the Window behaviour button, click on the Classes Tab, make sure derived is ticked and click on the New Class Methods button, click on the Insert Button.

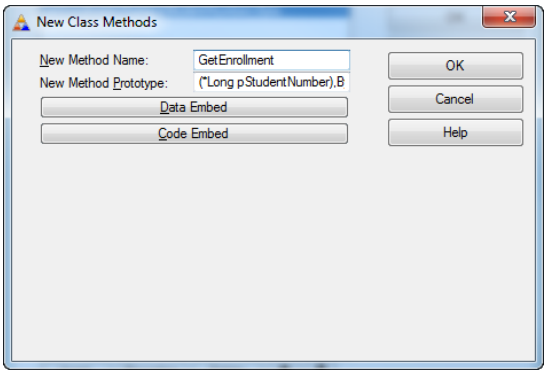


Figure 12. Adding a new method

Click on the code embed and add the following code to detect any student enrollment::

```
ENR:StudentNumber = pStudentNumber
ENR:ClassNumber   = 0
Set(ENR:StuSeq,ENR:StuSeq)
If Access:Enrollment.TryNext() = Level:Benign |
      and ENR:StudentNumber = pStudentNumber
    return true
end
return false
```

Goto the `SetQueueRecord` embed and add this code:

```
LOC:Enrollment = ThisWindow.GetEnrollment(STU:Number)
```

Change the Browse to have a color field indicating when the student is enrolled in at least one class.

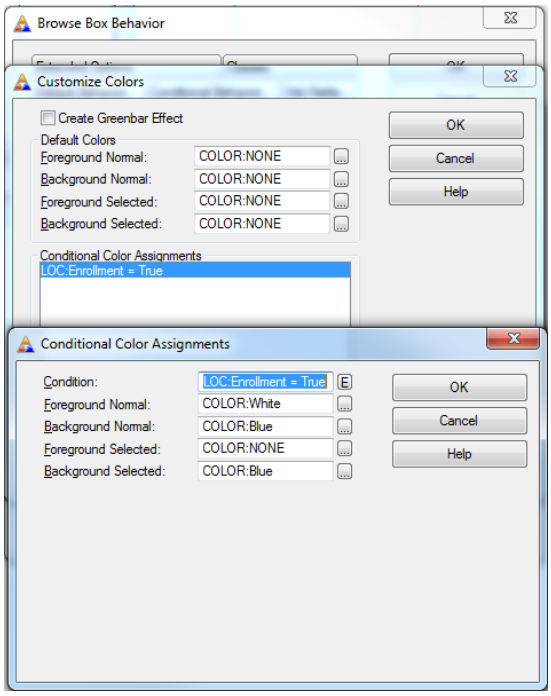


Figure 13. Conditional color assignments

Compile and run the app. When a student is enrolled in at least one class, the conditional color assignments will come into play (Figure 14).

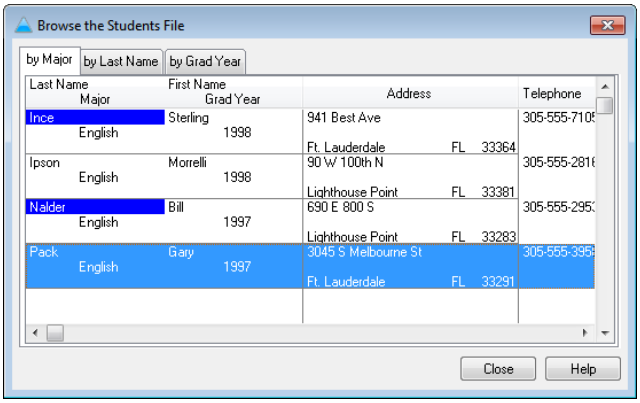


Figure 14. Highlighting students with enrollments.

Summary

In this article I showed how you can add your own methods into the `WindowManager` class to extend the class. As Steve has pointed out in his article this also helps you with prototyping the methods if you want to use them in your own class.

Download the source

Nardus Swanevelder was born and raised in South Africa. He was a networking engineer for seven years before he moved over to the commercial side of the business. Nardus has developed a Sale Cycle Management system for the Information and Communication Technology industry. He has been programming in Clarion since 1989, and holds B.Com and MBA degrees. In his spare time Nardus lectures Financial Management to B. Com Hons students at North-West University.

Article comments

BACK TO TOP