Clarion Magazine

Images, BLOBs, And The Imag'N OCX/DLL

Scott Daughtry uses the Imag'N OCX/DLL toolset from Pegasus Software to manipulate images in Clarion. In this article he describes how to store images in BLOBs and display those images on a browse.

Posted Tuesday, April 09, 2002

OCX Controls: Web Browser, Media Player, and pcAnywhere

OCXs are a quick and sometimes easy way to add a lot of functionality to your applications. Peter Rabolt shows how he used web browser, media player, and remote control OCXs to beef up one of his apps.

Posted Thursday, April 11, 2002

Detecting System Idle Time With Hooks (Part 1)

Faced with a need to detect system (not application) idle time, John Gorter turned to system hooks. System hooks are a way of plugging your own code into Windows' internal operations. There are a variety of system hooks available that let you do things like monitor Windows messages (a great way to determine system idle time), activate Computer Based Training applications, monitor low level keyboard and mouse events, and much more. Part 1 of 2.

Posted Friday, April 12, 2002

Trapping File Error Codes In The ABC ErrorManager

Faced with a need to detect at runtime when an application needed to convert a file to a newer format, Andrew goes looking for a way to trap file error codes in ABC. And he discovers that, contrary to popular opinion, it can be done.

Posted Monday, April 15, 2002

News

ExpressFlash Imports OE Address Book

INN Bio for 25-Apr-2002

Fomin Report Builder v.2.83

Free ASCII Reports Template

New ClarioNET(tm) Deployment Manager

New Icons Available From Gitano Software

Bug Poster Now Freeware!

ZipFlash 2.3 Released

etc-III Is Just Three Weeks Away

Imaging Template Update/New Imaging Product

New ImageMan Templates Answers The XP Question

Icetips Magic Buttons

RADventure Releases SQL Browse

SoftMasters Toolbar Controls v1.0 Released

Detecting System Idle Time With Hooks (Part 2)

Faced with a need to detect system (not application) idle time, John Gorter turned to system hooks. System hooks are a way of plugging your own code into Windows' internal operations. There are a variety of system hooks available that let you do things like monitor Windows messages (a great way to determine system idle time), activate Computer Based Training applications, monitor low level keyboard and mouse events, and much more. Part 2 of 2.

Posted Wednesday, April 17, 2002

Data Structures and Algorithms Part I: The Dynamic Stack

On the newsgroups it happens time and time again: a Clarion newbie asks: "How do I implement a Tree in Clarion?" or "How do I implement a List in Clarion?" These aren't questions about tree controls or list boxes; they're about established, standard data structures. Here is the first article in a series by Alison Neal outlining how to implement standard data structures and algorithms in Clarion.

Posted Friday, April 19, 2002

Search Clarion Magazine's News Archive

You've used Clarion Magazine's search engine to track down articles. Now you can also search our news archive. This is a gold mine of product announcements and other items of interest to Clarion developers. This index is available to anyone with a free membership or subscription.

Posted Monday, April 22, 2002

It's an XML World!

XML is everywhere, as Jim Kane is beginning to see. In this article Jim shows how to use the MS DOM XML parser to manage XML documents.

Posted Thursday, April 25, 2002

Required Fields Times Two

A lot has been written lately in Clarion Magazine about validating required fields. Carl Barnes validates the notion that there's no such thing as too much Clarion knowledge,

and lays out his approach to required field handling.

Posted Friday, April 26, 2002

SoftVelocity's Upcoming Product Technology - Abstract

Clarion Magazine's Reviews Editor, Tom Hebenstreit, recently attended SoftVelocity's Technology Workshop in Florida, and came away stoked about the company's soon-to-be-released product and its future direction. This freely available abstract outlines the topics covered in Tom's report.

Posted Monday, April 29, 2002

SoftVelocity's Upcoming Product Technology (Part 1)

Part 1 of Tom Hebenstreit's report on the SV Technology Workshop covers topics such as legacy support, true threading, and ADO/SQL support.

Posted Monday, April 29, 2002

SoftVelocity's Upcoming Product Technology (Part 2)

Part 2 of Tom Hebenstreit's report on the SV Technology Workshop covers topics such as the Clarion/ASP templates, PHP templates, XML support, and a wide range of future product development possibilities.

Posted Monday, April 29, 2002

Looking for more? Check out the **site index**, or **search the back issues**. This site now contains over a million words of Clarion-related information.

EmailReport 2.2 Saves Reports To Disk

BigTamer(tm) Templates & Utilities Add C4 Compatibility

Gitano's gFileFind Shareware

SealSoft Distributor Discount

twMySQL 1.0 Released

gCal 3.0 Update

twTools 1.3 Released

GUI.IT Beta Released (Now 0-HaZzle)

EasyMultiTag 2.00 Released

INN Bio for 28-Mar-2002

RPM 5T for C55G Re-Released

PostNet Barcodes For C55G Available

Search the news archive

Images, BLOBs, And The Imag'N OCX/DLL

by Scott Daughtry

Published 2002-04-09

The Imag'N OCX/DLL toolset from Pegasus Software provides robust image file manipulation. An application I'm currently writing uses several BLOB fields to store *.jpg image files, and I wanted the listbox to use the images similar to hot fields (i.e. when the highlight bar is moved, the images displayed onscreen refresh automatically). The task of displaying image thumbnails without slowing down listbox speed provided an excellent opportunity to reacquaint myself with this excellent programmer's library. In this article I'll cover BLOB image handling, as well as the specifics of using the Imag'N tools with Clarion.

Dictionary Setup

To use BLOB fields to store images (or other binary data) you can use the Topspeed (TPS) file driver. You can define up to 255 BLOB fields in a single TPS file; the dictionary also lets you specify if the BLOB field will store binary data or not. Figure 1 shows the settings I used in my application.

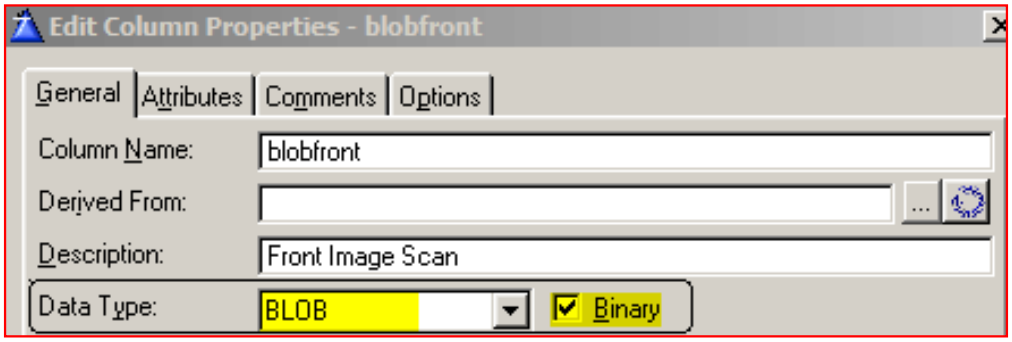

Figure 1. Setting the BLOB data type for a file field.

After defining the BLOB fields in the dictionary I also defined a single binary MEMO

field that is 10 characters in length, immediately after the last BLOB:

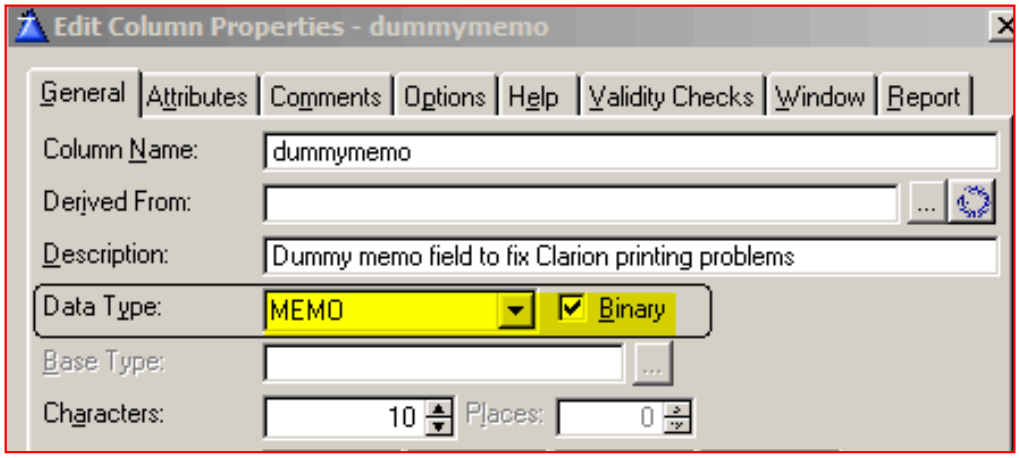

Figure 2. Creating the dummy memo field.

This fixes a memory allocation problem with printing BLOB fields. You only need to define one "dummy" memo field in the dictionary.

Import these procedures!

The demo application supplied with the ImagN templates has several useful procedures that need to be imported into your application. Do the following:

- Open your *.APP file
- Click File, then Import From Application
- Open the IMW5.APP application (for ImagN 4.0 users; if you are an ImagN 3.x user you will open the file IMW.APP). -You can ignore warnings about dictionary differences between the two applications.
- Select the procedures highlighted in Figure (OpenImageCB, ReadImageCB and SaveToBlob):

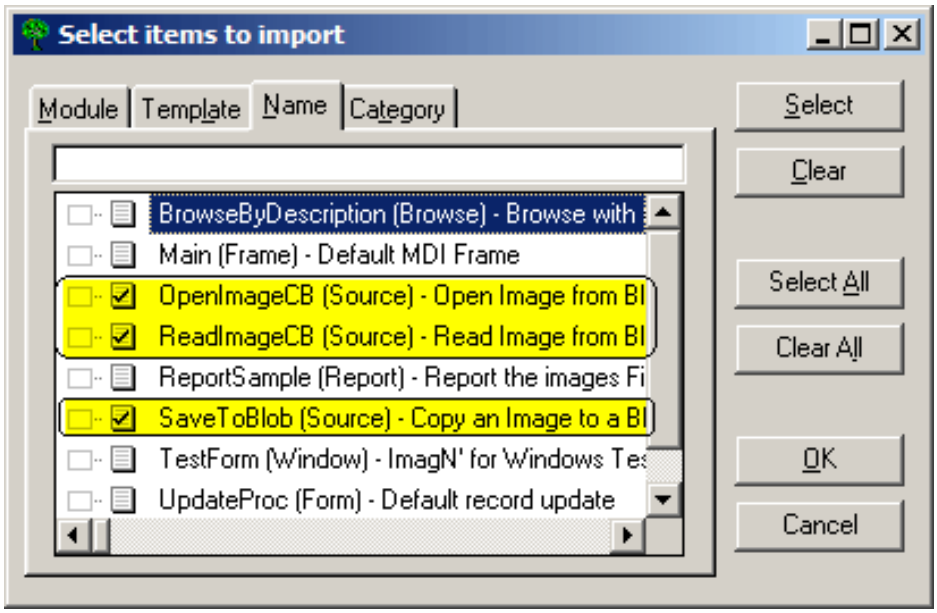

Figure 3. Procedures to import into your application

- Click OK to import the procedures into your application.

Here's an explanation of each of these procedures:

`OpenImageCB` - Returns the size of the `BLOB` field contents; you will need to edit the source code for this procedure to use the field name(s) for your application. For example, I have two `BLOB` fields that I am displaying on the browse procedure (`DIS:BlobFront` and `DIS:BlobBack`), so my source code looks like this:

```
! Retrieve size info; 0=DIS:blobfront, 1=DIS:blobback
if (Instance = 0) then return DIS:blobfront{prop:size}.
if (Instance = 1) then return DIS:blobback{prop:size}.
```

Remember that the `IF` statement cannot be in position one of the source - if the editor highlights your code in red, move it over one space!

`ReadImageCB` - Returns offset information for ImagN for each of the `BLOB` field contents; you will need to edit the source code for this procedure to use the field name(s) for your application. For example, I have two `BLOB` fields that I am displaying on the browse procedure (`DIS:BlobFront` and `DIS:BlobBack`), so my source code is as follows:

```
! Return read amounts per field;
! 0=DIS:BlobFront, 1=DIS:BlobBack
! Instance 0=front image
if (Instance = 0) then wrk[1:ReadAmt] |
  = DIS:blobfront[ReadOffset : ReadOffset + ReadAmt ].
! Instance 1=back image
```

```
if (Instance = 1) then wrk[1:ReadAmt] |
  = DIS:blobback[ReadOffset : ReadOffset    + ReadAmt ].
! Copy image to RAM
IM_CopyMem(Dest, Wrk, ReadAmt)
return ReadAmt
```

Remember, only the comment lines can start in position one!

`SaveToBlob` - If you want to provide `BLOB` content export capability for your users this procedure will use low-level functions to write the file to disk. It isn't needed for the browse screen display, but may be handy later.

The ImagN control template

Open the browse procedure that you want to display the BLOB images in the window formatter screen. Reduce the size of the listbox to free up space to display thumbnail sized versions of the BLOB images. Drop the ImagN control template onto the window (`Populate -> Control Template -> ImagN - ImagN' for Windows Template`), and then resize it to taste. You can define the properties of the template by right-clicking on the newly populated ImagN region, or from the Extensions button on the procedure properties screen. Figure 4 shows an example configuration, with explanatory notes following.

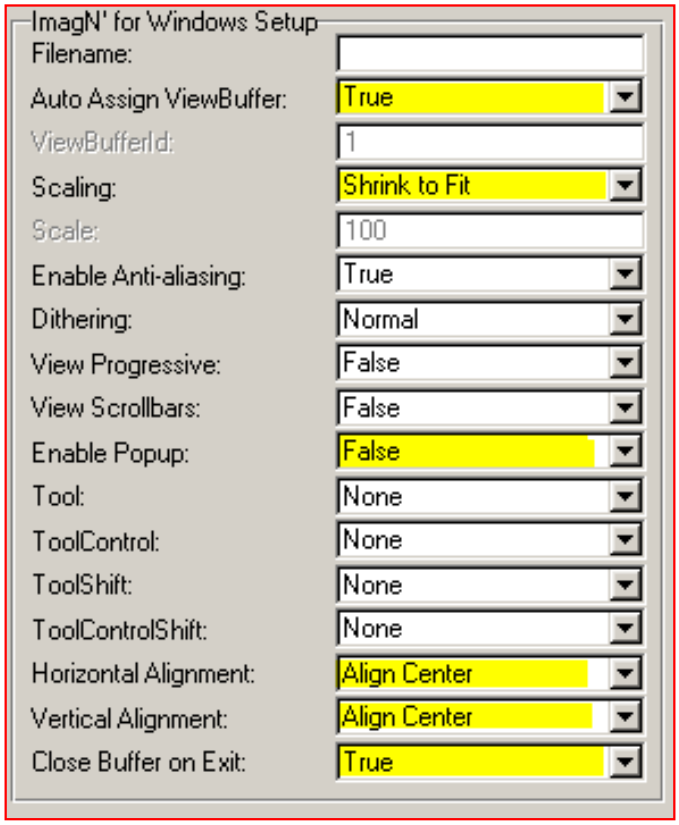

Figure 4. An example Imag'N configuration

Field	Setting
Auto Assign ViewBuffer	ImagN will take care of creating pointers to each image internally if set to TRUE
Shrink To Fit	Take care of images that are larger than the control template region area
Enable Popup	Turned off to disable the right click menu that accesses ImagN options for image display features (zoom, rotate, etc)
Horizontal Alignment	I wanted the image centered onscreen in the region area
Vertical Alignment	Same as above
Close Buffer On Exit	Allow ImagN to perform housekeeping to free up memory buffer area's when the browse procedure is closed

Assign procedures to the browse

Just because you add a procedure to an application doesn't automatically mean you can call it from anywhere. You either have to declare the procedure globally (using the Declare Globally checkbox on the procedure properties window) or add the procedure to the calling procedure's list. To do the latter, click on the Procedures button for the browse procedure and add two of the three imported ImagN procedures, as shown in Figure 5:

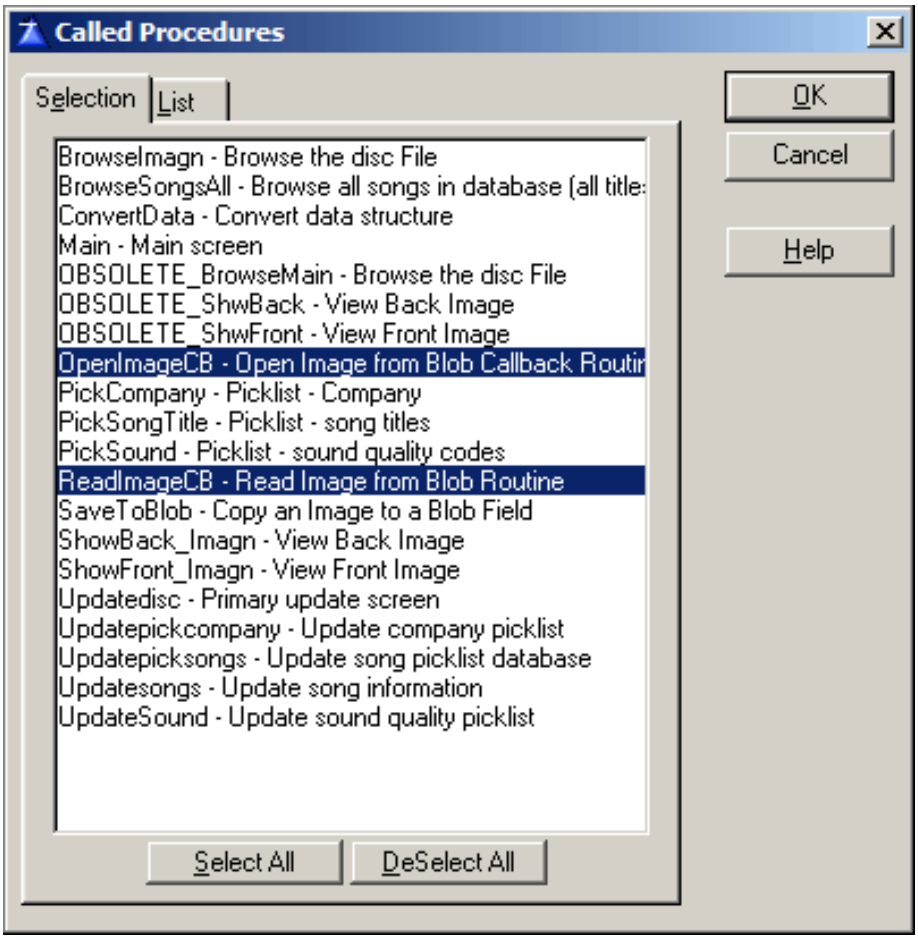

Figure 5. Adding the Imag'N procedures to the called procedure list

Click OK to save and exit. This enables the browse procedure to use the `OpenImageCB` and `ReadImageCB` callback functions that the ImagN control template will require to properly display the BLOB images.

Embed the listbox code

Two embeds are used to ensure the listbox works correctly with the ImagN control templates:

`Control Event Handling, after generated code, Accepted` - this embed is used when the left mouse button is clicked on a listbox entry (i.e. selecting a listbox record)

`Control Event Handling, before generated code, NewSelection` - this embed is used when the numeric keypad is used to move the listbox highlight bar (i.e. up and down arrow keys)

Open the browse procedure again into the Window Formatter and double-click on the listbox to display available embeds. Click the Legacy Embeds button (red/yellow

Clarion icon), then scroll down the list to the first embed that needs some source code added (Control Event Handling, after generated code, Accepted). Click the Insert button, and then source. In the source window you will add in the code to perform a GET on the database record, and then call the ReadImageCB procedure. Some example code looks like this:

```
! Read the current highlighted listbox record
GET(DISC,DIS:Keytitle)
! Display the first BLOB image
!  (front of cd - instance 0)
IMW:RetCode = IM_ReadImageCB(?ImagN:ViewBufferId,0, |
  Address(OpenImageCB), Address(ReadImageCB),|
  0, 0)
! Display the second BLOB
! (back of CD - instance 1)
IMW:RetCode2 = IM_ReadImageCB(?ImagN:2:ViewBufferId,|
  0, Address(OpenImageCB),    |
  Address(ReadImageCB), 0, 1)
```

The IM_ReadImageCB procedure has six parameters, but only two (the first and last) are important to you. The first parameter is the USE label for the control template you placed on the procedure window; the last parameter is the INSTANCE integer (0=first control template on the window, 1=second control template on the window, etc).

Copy the source code you just typed to the Windows clipboard, then save the embed source code. Open the second embed (Control Event Handling, before generated code). Click the Insert button, and then Source. In the source window paste the exact same source code, and then save it.

Local data

You need to add a LONG data field for each BLOB image you will be displaying on the listbox window - it will contain the return value of the IM_ReadImageCB procedure. In the example embed code above I have IMW:RetCode and IMW:RetCode2 variables defined that are used for this purpose.

You're ready to compile!

After double checking embed code compile the application; the finished product will look something like this:

Figure 6. The finished product.

ImagN is so fast in displaying images that the screen refresh is instantaneous when the listbox highlight bar is moved; using standard Clarion image controls the refresh sometimes took 3-10 seconds! The image quality was also greatly improved, as ImagN allows for image dithering and other image optimization tricks.

Summary

ImagN provides Clarion programmers with a powerful image manipulation toolkit that adds features and sales value to your application. With very little embed code a listbox can display stored data as quickly as the numeric keypad is depressed. The Imag'N template set is available from Pegasus Software for $399.

Scott Daughtry is an active duty Air Force information technology manager, and has been a Clarion user since 1996. He primarily writes automation applications for the service, but dabbles in shareware ideas for his web site, www.sdaughtry.com. He is married to the former Jacqueline Hilton from Pampa, Texas. Scott and Jacqueline have three children: Jessica, Amy and Joshua.

Reader Comments

Add a comment

My recollection is that this package does not give you...
Not quite true; if the sales of your application are 250K...
How well does Imag'N work for printing photos in the...
Printing works extremely well. I've not found a single...

OCX Controls: Web Browser, Media Player, and pcAnywhere

by Peter Rabolt

Published 2002-04-11

If you search the Internet for "OCX", you will find there are numerous OCX (formerly called ActiveX) controls available (some are freeware) that can help you to interface your software with all kinds of devices and technologies. By using an OCX control, you add its features to your application and enhance the capabilities of your software.

In its simplest form, you provide data to the OCX control by configuring its properties and methods. In a more complex form, you can trap events triggered by the control using callback functions (in which you create a function that the OCX can call), but this is a detailed process that goes beyond the scope of this article. For more on callbacks see Morten Thomsen's Clarion Online article Getting Started with OCXs. Morton's article discusses Clarion 4 and is a little outdated, but still provides useful background.

I recently worked on a project that required playing video files streaming over the Internet. It also required the ability to connect to workstations over the Internet to help resolve day-to-day support issues, and also to display the company intranet. In doing research for this project, I realized I could have purchased some of the many templates available from third party vendors to handle my requirements, but I also found there were OCX controls available to help accomplish my project. I felt this was a chance for me to get some real world experience in setting up, configuring, and communicating with OCX controls.

To include the web browser capability in my project, I followed the process described by Matt Grossmith in his ClarionMag article Using The Web Browser OCX. This article provides an easy step-by-step tutorial on how to add web browsing

capability to your application.

To include the video playback capability in my project, I investigated the Windows Media 6.4 control. After reviewing MS newsgroups and MSDN, I compiled a list of available properties and methods along with descriptions of each option (see Table 1 below).

To include the remote connectivity to an existing pcAnywhere host, I investigated how I could call pcAnywhere from my application. The best I could do was call pcAnywhere and let it take over the screen until the user was finished. This left my application with no control during the pcAnywhere session. Later that week I read a post in one of the Symantec newsgroups about the pcAnywhere ActiveX control. After making further inquiries, I found that there was in fact a remote control OCX available. You can download the control from the Symantec web site.

Download the control into your Windows system folder, and then follow the instructions below.

Installing the OCX

To register the control with the Windows 98 operating system, open a DOS window and proceed to the Windows system folder. At the prompt, type:

```
REGSVR32 [OCXCONTROL.OCX]
```

Replace the [OCXCONTROL.OCX} with the name of the ocx file. You need to register the control on your development system as well as on any other system that will be using your application.

> **NOTE:** Registering the control with other versions of Windows may be slightly different, so it is best to review your operating system documentation.

To use the control in Clarion 5.5, open a new or existing window in your app and place an OLE/OCX control on it. On the properties tab of the control, enter a Use variable, such as `?Ocx1`. Check the 32-bit checkbox, and set the control type to OCX.

For the Windows Media OCX, select Windows Media Player from the Object type drop down list. This type will show as `MediaPlayer.MediaPlayer.1`. If you see an entry in the list that shows `WMPlayer.7`, *do not* use this control – it is for the media

player version 7 control which has completely different properties and methods, and does not have as many detailed options as the earlier version 6.4 control.

For the pcAnywhere Remote OCX, select `pcAnywhere Remote` from the Object type drop down list. This type will show as `pcAnywhere.Remote.1`. Figure 1 shows an example of the pcAnywhere OCX properties screen.

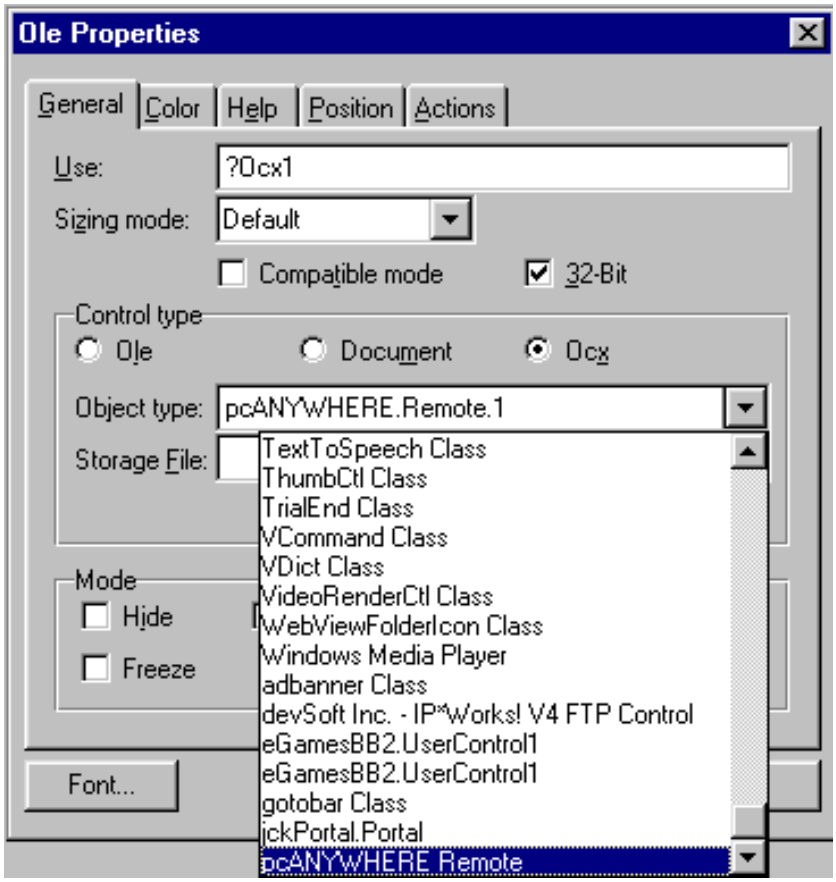

Figure 1. The pcAnywhere remote OCX properties window

The last step is to configure the properties and methods of the control you are using. I selected the embed point After Opening the Window to put my ocx setup code, since I wanted the control to start immediately.

The syntax to set a property is this:

```
?Ocx1{'property'}=value
```

To set a method you do the following:

```
?Ocx1{'method(param1,param2…)'}
```

Windows Media control properties and methods

Table 1 shows the properties/methods for the Windows Media Player OCX, along with the settings I used. These settings enabled me to modify characteristics of the media player, such as display size, volume, background/foreground color and toolbars.

Property/Method	**My Setting**	**Description**
AutoSize	-1	Used with DisplaySize
AutoStart	-1	Don't wait for user confirmation
AllowChangeDisplaySize	0	User can't change video size
AutoRewind	0	Do not auto-rewind at end of video
Balance	0	0=equal -10000=all left speaker, 10000=all right speaker
ClickToPlay	0	Let user click image to start and stop
DisplayBackColor	0	Background color
DisplayForeColor	16777215	Foreground color
DisplaySize	4	0=reg 1=half 2=double 3=full 4=fit 5=1/16 6=1/4 7=1/2
Enabled	-1	Enable all user input
EnablePositionControls	-1	Enable skipahead skipback fastforward fastreverse
EnableFullScreenControls	0	Allow controls in full screen mode
EnableTracker	-1	Enable track bar
Mute	0	0=not muted -1=muted
PlayCount	1	Number of times to play
Rate	1	Playback rate .5=slower 1.5=faster
ShowControls	-1	Show all controls
ShowAudioControls	-1	Show speaker volume and mute
ShowPositionControls	-1	Show skipahead skipback fastforward fastreverse
ShowTracker	-1	Show progress trakker
TransparentAtStart	0	0=not transparent

VideoBorderWidth	5	Video border width in pixels
VideoBorderColor	255	Video border color
VideoBorder3D	0	0= no video border 3d effect
Volume	600	1-10000=lower vol 0=full volume
FileName		Filename of local or streaming url

Table 1. The Windows Media Player OCX properties/methods

pcAnywhere Remote properties and methods

Table 2 shows the properties/methods for the pcAnywhere OCX, along with the settings I used. These settings enabled me to modify remote characteristics, such as display size, colorscale, cache size and toolbars.

Property/Method	My Setting	Description
Colorscale	0	Number of remote displayed colors
EZScrollHotKey	0	Special keys to scroll like mouse
CacheFileSize	15	Default cache size
LockHostKeyboard	0	Allow host/remote keyboard
BlankHostScreen	0	Do not blank host screen
ReduceHostDesktop	2	Reduce host to remote resolution
SyncMouseButtons	1	Sync host/remote mouse clicks
TrackHostWindow	1	Sync host/remote windows
TrackHostMouse	1	Sync host/remote mouse
StartEZZoom	0	Start special EZZoom mode Select section of screen to zoom
StartFullScreen	0	Stay within application window
ShowToolBar	1	Show remote toolbar
ShowScroolBars	1	Enable scroll bars

Table 2. The pcAnywhere OCX properties/methods

I found that properties are usually setup at the start of the procedure such as "after opening the window" embed point, whereas methods are usually the result of user input coded using a button control:

To control the Refresh() pcAnywhere method, I created a button and titled it Refresh. In the Completed Embed of the button I put the following code:

```
Refresh()
```

When the user presses this button, the pcAnywhere screen is refreshed. To control the Disconnect() pcAnywhere method, I created a button and titled it Disconnect. In the Completed Embed of the button I put the following code:

```
Disconnect()
```

When the user presses this button, the pcAnywhere remote disconnects from the host.

To control the pcAnywhere Connect()method, I created a button and titled it Connect. In the Completed Embed of the button I put the following code:

```
Connect(param1)
```

When the user presses this button, the pcAnywhere remote connects to the host specified by param1, or prompts for a host if you leave the param1 spot empty.

Summary

Using these three ocx controls actually turned this project into a success, and in the end I learned another Clarion feature I could apply to future projects. In today's exploding software industry, purchasing pre-written routines or interfacing with pre-existing technologies allows the developer to produce a more robust product in less time while maintaining a very competitive retail price.

Download the source

Peter Rabolt grew up in Brooklyn, New York and has dabbled in computers since 1978 when he received a Radio Shack TRS-80 Model 1 with 16k ram for his 16th birthday. Peter taught himself how to program, since at that time his high school had no computer courses. After graduating college, Peter aspired to run his own computer company. Today Peter owns Synchronized Systems, a Long Island New York-based

firm that provides customized software, hardware, and networking services. Peter has used Clarion since the days of CPD 2.1 and still supports systems in 2.1, CFD 3.1, as well as CW.

Reader Comments

Add a comment

I've added a link to Peter's example application. Also from...

Detecting System Idle Time With Hooks (Part 1)

by John Gorter

Published 2002-04-12

Sometimes an application needs to know how long the system hasn't had input, which is my definition of system idle time. Facing this problem in one of my applications had me investigate a little further and made me come up with some information which might come in handy for other similar situations we Clarioneers face so now and then. The key was to use Windows system hooks. System hooks are a way of plugging your own code into Windows' internal operations. There are a variety of system hooks available that let you do things like monitor Windows messages (a great way to determine system idle time), activate Computer Based Training applications, monitor low level keyboard and mouse events, and much more.

In this article I'll show how to use window hooks and file mappings to detect this system idle time. File mappings are a way for two processes to share a piece of memory without having to create a file, but more on that later.

System hooks

In Windows 2000 there is an API call that returns the system idle time. This is a new function with the following prototype:

```
LASTINPUTINFO lii ;
lii.cbsize = sizeof (lii) ;
GetLastInputInfo (&lii) ;
```

Unfortunately, this function only exists on Windows 2000 and since most of my tools need to support Windows 95/98 as well, this call wasn't an option. So how did I do it?

The key thing to do is install a system-wide hook, which, as I mentioned earlier, is a callback function that lets you monitor interesting events, such as input events.

From MSDN :

> A hook is a point in the system message-handling mechanism where an application can install a subroutine to monitor the message traffic in the system and process certain types of messages before they reach the target window procedure.

This is exactly what I want, to know when input occurs. Hooks are far more powerful than the way I use them here, however. You can, for instance, monitor keystrokes, discard them and send others. You are in control of who gets which message...but I'm getting ahead of myself.

It's worth mentioning that the hook functions need to be placed in a DLL which is loaded for every desktop application that is calling the hook functions. Second, I advise you to use the hook mechanism only if there is not a cleaner way to do things, since a system-wide hook has quite an impact on the overall system performance. In this example, if you are only installing on Windows 2000, use the `GetLastInputInfo` API call.

Why not just subclass?

This article does not provide a detailed specification of Windows' subclassing capabilities. I assume that you already know what subclassing is; if not there are several articles you can read for background. By way of review, the technique mainly involves replacing the default wndfunction for the Window with a custom function. This lets my function handle all the incoming messages and act on them as I plan. This might sound like something useful, but there are shortcomings with subclassing that make the needed functionality practically impossible.

The first shortcoming is the fact that subclassing requires parameters based on user space addresses. To set up a subclassing function I have to pass a function address. This works perfectly well if the window being subclassed is in my address space. If, however, the window to be subclassed is the address space of another process, things go terribly wrong. It is highly unlikely that the process the window to be subclassed is in will have the right function at that address. It probably hasn't got the function at all. The fact that I cannot subclass windows from other processes eliminates this technique as an option to detect system idle time.

The second shortcoming is that even if it were possible to subclass over process boundaries, I'd have to go through all the windows in the system and subclass all of them, not to mention the fact that windows can be destroyed afterwards and new ones can be created. I'd have to detect those activities and respond accordingly. These shortcomings eliminate subclassing as a viable technique.

With hooks, however, I do not have to worry about window deletion and creation; all input messages are sent to my function regardless of which Window had input focus at the time the input occurred.

Which messages to catch?

In detecting input activity I am interested in the mouse and the keyboard messages, since those devices are responsible for all the input messages generated by the system. I can install hooks for those messages using the function `SetWindowsHookex`, which takes a first parameter which implies the type of hook I want to install. (There are quite a few hooks available: for a complete list see the MSDN web site.

Other parameters to the function include a callback function I provide for handling the incoming message, the handle to the DLL containing the hook procedure pointed to by the callback parameter, and the identifier of the thread with which the hook procedure is to be associated (this can be null). The function returns a handle to the hook which my callback function needs to call when it is through working with the message.

Once the hook is installed, that is, when the call to `SetWindowsHookEx` succeeds, Windows will load the DLL in which the hook-function resides. Windows does this for the current application whenever an input event occurs on the window that has the input focus (if applicable - that is, if the DLL wasn't already loaded on a earlier occasion).

Thus the call to set up the system hook could look something like this:

```
HHOOK hKBHook = SetWindowsHookEx
   (WM_KEYBOARD, MyKeyboardHook, hInst, NULL) ;
HHOOK hMouseHook = SetWindowsHookEx
   (WM_KEYBOARD, MyKeyboardHook, hInst, NULL)    ;
```

However, this is the C(++) version, which brings me to the next problem: variable sharing across process boundaries. If I want to keep track of the elapsed time after

the last input event, I could use the API GetTickcount() to fill a variable with the difference between the last and the current tickcount, but since every application has its own address space, the variable would have an instance in each address space as well, resulting each application just detecting its own idle time, which is not what I intended.

> **NOTE**: There are WH_KEYBOARD_LL and WH_MOUSE_LL hooks that let functions detect low level input from these devices. These hooks do work because these hook are called in the context of the one thread that installed them, but unfortunately they also aren't supported by Windows 98, only Windows NT and Windows 2000.

In C++, I can do something like this to create shared data:

```
#pragma data_seg("MYDATA")
HHOOK hKeyboard = NULL ; // must be an initialized data
                         // else it does not create
                         // a new section
HHOOK hMouse = NULL ;    // must be an initialized data
                         // else it does not create
                         // a new section
DWORD dwCount = 0 ;      // must be an initialized data
                         // else it does not create
                         // a new section
#pragma data_seg ()
#pragma comment (linker, "/section:MYDATA,rws")
```

This fragment of code tells the linker to generate a new data section and arm it with the RWS (Read Write Share) attributes.

The share attribute tells the system to map a view to this piece of data in each instance of the running application, not the data itself. This causes all instances to physically use the same global variable(s). In Clarion however, I cannot seem to get the right pragmas set for the job (if someone does know how to do this please tell me), so that leaves me to implement this functionality myself using the technique of file mappings.

File whattings?

File mappings can be used to share a piece of memory between two processes; this is exactly what I want, since I need a place to store my system-wide variable. To implement this behavior I use the API call CreateFileMapping with a first variable of INVALID_HANDLE_VALUE (which equates to -1). This value tells the system to create a file in memory, not an actual disk file. The last parameter is reserved for an

`ObjectName`. Object names make it possible to locate objects over process boundaries; to use the object, you just look for its unique name.

`CreateFileMapping` returns a handle to a kernel object which may or may not already exist (the object will be created if necessary). I can check for this through the `GetLastError()` function, which returns `ERROR_ALREADY_EXISTS` when I try to create an already created object.

Next I have to call `MapViewOfFile` to map a view of the file in my address space. I supply the handle to the `FileMapping` object and tell the function which access rights I want to have with the memory. The last three parameters are usually set to 0, telling the function not to take an offset in the file and map the whole thing in the application's address space. In this case I only have one long variable mapped. Figure 1 shows a graphical presentation of the FileMapping architecture.

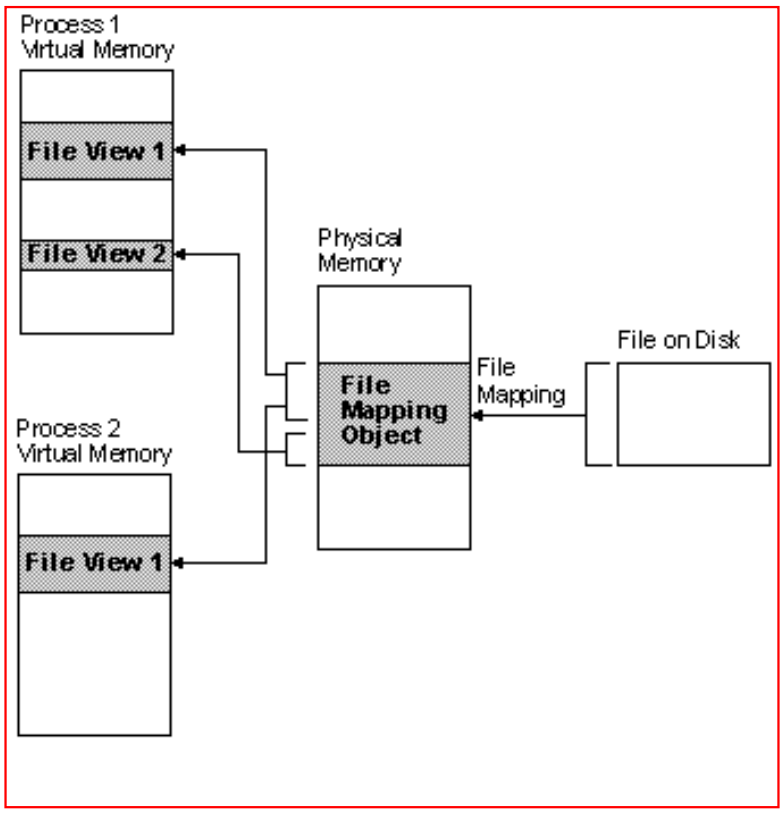

Figure 1. The FileMapping architecture.

As you can see, two processes, `Process1` and `Process2`, both have a handle to the same `FileMapping` object, which is in turn taking care of the actual content of the disk file or, in case the `CreateFileMapping` function is called with a first parameter of `INVALID_HANDLE_VALUE`, a piece of memory backed by the operating-system paging file.

I now have all the pieces I need to proceed with my system hook. Next week I'll show how to implement the system hook in a Clarion application.

Download the source

John Gorter has been programming in Clarion for three years, before which he studied business informatics. He has just passed the MCSD exams and is now busy creating web applications with C55 Internet Connect. John lives in the Netherlands, and when not programming, reads about programming.

Reader Comments

Add a comment

This file mapping beast is of particular interest! How can...
Ok to test if it already existed GetLastError returns...

Trapping File Error Codes In The ABC ErrorManager

by Andrew Guidroz II

Published 2002-04-15

It finally happened. I needed to change two file layouts at various locations. Some of the locations already had a newer version of the program with one of the file layouts changed. Some did not. So, how could I test for the need for a conversion with ABC?

Running my software without the correct file layout would result in the infamous Invalid Record Declaration message. And then the program would halt. I didn't want this error to appear. I wanted to trap it and then convert my file.

What do you mean, it can't be done?

Within ABC, file methods can return various results. The problem becomes trying to identify the exact error that triggered a problem. `ERRORCODE()` and `FILEERRORCODE()` does not always contain the value you need: the classes can clear them out. So I needed to override some of the standard ABC behavior. Fortunately that's easy to do.

The first step was making sure the error didn't appear. Digging in the `aberror.inc` and `aberror.clw` files (the `ErrorManager` classes), I found the `Silent` property, declared like this:

```
GlobalErrors.Silent = TRUE
```

This property tells ABC whether or not to display a message box with the error. I started to feel pretty good about my prospects. I set `GlobalErrors.Silent` to `False` and started testing. And then I hit the wall.

The program kept HALTing. This was due to the fact that Msg:OpenFailed is a Level:Fatal message. The ErrorManager allows various levels for certain errors, and will ignore some error levels entirely. Other errors are handled, but are not fatal. So I added the following bit of code, which basically tells the ErrorManager how seriously to take an open failure:

```
GlobalErrors.SetFatality(Msg:OpenFailed, Level:Notify)
```

Now an error wasn't displayed but neither was it totally ignored: I could test for it. But I still didn't know exactly what the error was. ABC will only return the severity of the level, not the actual ErrorCode() itself.

ABC has access to error codes, error messages, and all sorts of wonderful information that is private and protected. You can't get at it from just any old place. I was totally frustrated because there was all the information I needed just sitting on that error window that it once displayed and I couldn't grab it. And that's when the light bulb went on.

I realized I really don't need the ErrorCode() at all: I just need the textual error information. Perusing the ABC classes showed me a method of the ErrorClass called SubsString. This method takes a standard string that describes what the error message should be and replaces some symbols with the ErrorCode() and Error(). The real beauty of it is that it returns the completed string. With that string, I could parse out the ErrorCode() easily because the ErrorCode() is contained within a pair of parenthesis. All I need to do is the following:

```
MyString = GlobalErrors.SubsString()
```

Right? *Wrong!*

SubsString is a protected method. This means it can only be called from within the ErrorClass. Or a class derived from ErrorClass. Bingo!

The following is my new SnowErr.INC file that I place in the LIBSRC directory of Clarion 5.5 ...

```
   !ABCIncludeFile

OMIT('_EndOfInclude_',_SnowErrPresent_)
   _SnowErrPresent_ EQUATE(1)
   include('ABError.INC'),ONCE

ErrorManagerForSnowDog CLASS(ErrorClass),TYPE,|
    MODULE('SnowErr.CLW'),LINK('SnowErr.CLW',_ABCLinkMode_)|
```

```
        ,DLL(_ABCDLLMode_)
    SnowDogReturnErrorString PROCEDURE(),STRING,VIRTUAL
    SnowDogReturnLastErrorCode PROCEDURE(),LONG,VIRTUAL
    END

!_EndOfInclude_
```

I've defined two methods. One returns the entire error string and one will return only the `ErrorCode()` that I need. Here is the `SnowErr.CLW` file in my `LIBSRC` directory.

```
    MEMBER
    MAP
    END
    INCLUDE('SnowErr.INC'),ONCE

    ErrorManagerForSnowDog.SnowDogReturnErrorString PROCEDURE()
    CODE
    Return Self.SubsString()

ErrorManagerForSnowDog.SnowDogReturnLastErrorCode PROCEDURE()

CurrentSubsString CSTRING(2000)
BeginPosition SHORT
EndPosition SHORT
ReturnValue LONG

    CODE
    CLEAR(ReturnValue)
    CurrentSubsString = Self.SubsString()
    BeginPosition = INSTRING('(',CurrentSubsString,1,1)
    IF BeginPosition
      BeginPosition += 1
      EndPosition = INSTRING(')',CurrentSubsString,1,BeginPosition)
      IF EndPosition
        EndPosition -= 1
        ReturnValue = CurrentSubsString [ BeginPosition : EndPosition ]
      END
    END
    Return ReturnValue
```

Put both files in your ABC `LibSrc` directory and press the Refresh Application Base Classes Information button found in Global Properties, on the Classes tab.

The first method returns the entire string. That's nice, but not terribly useful in tight code. The second method gets the actual `ErrorCode()`. It knows that all error codes are surrounded by a parenthesis pair, and it strips that out of the string and returns it.

Using the class

To use the `ErrorManagerForSnowDog` class, place the `.CLW` and `.INC` files in your `LIBSRC` directory. Then go to `Global Properties` for your application, then to the

Global Objects tab. Click the ErrorManager button and uncheck the Use Default ABC Class. Select ErrorManagerForSnowDog as the new Base Class. Now, the application error handling code becomes:

```
GlobalErrors.SetFatality(Msg:OpenFailed, Level:Notify)
GlobalErrors.Silent = TRUE
IF Access:CCIOperator.TryOpen()
   ! open failed
   IF GlobalErrors.SnowDogReturnLastErrorCode() = 47
      ! MESSAGE('The open failed for srep --- convert time')
      ProcessConvertV1CCIOperatorToCurrentVersion
   END
ELSE
   IF Access:CCIOperator.UseFile()
      IF GlobalErrors.SnowDogReturnLastErrorCode() = 47
         ! MESSAGE('The use file failed for srep --- convert time')
         ProcessConvertV1CCIOperatorToCurrentVersion
      END
   ELSE
      ! Message('The open was successful')
   END
END
GlobalErrors.SetFatality(Msg:OpenFailed, Level:Fatal)
GlobalErrors.Silent = FALSE
Access:CCIOperator.Close
```

Of course you'll want to replace the ProcessConvertV1CCIOperatorToCurrentVersion (Cajun file naming convention!) procedure with your own conversion procedure (or whatever code you like). The reason for testing for error code 47 (invalid record layout) after both the TryOpen and the UseFile is because the file may have LazyOpen set to TRUE. The TryOpen may not find out there is a problem until it hits the UseFile.

By making this new ErrorClass my default error class for all my applications, I now have access to the ErrorCode() anywhere I like. Yes, it really can be done!

Download the source

Andrew Guidroz II, when he isn't traveling around the countryside watching his 2001 SEC Champion LSU Fighting Tigers, writes software for all facets of the insurance industry. His famous Cajun cookouts have become a central feature of Clarion conferences throughout the U.S. Andrew's Cajun website is www.coonass.com.

Reader Comments

Add a comment

Excellent article - I only wish you had written it six...
Thank you for this article, Andrew! I often need (and...
Andrew - Excellent article! You've solved a mystery...

Clarion Magazine

Clarion News

Published 2001-11-21

ExpressFlash Imports OE Address Book

Sterling Software has released ExpressFlash 2.1, which now imports all fields from the Outlook contacts address book. Apart from this, ExpressFlash 2.1 will also read messages from email & newsgroups in Outlook Express & Outlook 2002/2002. Demo available.

Posted Monday, April 29, 2002

INN Bio for 25-Apr-2002

This week, the Icetips News Network is pleased to present another first, a bio from South Africa. This gentleman is rather well-known, and has written/been involved with some of the most popular third-party tools. He regularly attends and presents at conferences around the world, and we'll see him leading a class on ABC at ETC. For now, see what he says on programming, templates, business, and being a witness to his country's recent history.

Posted Monday, April 29, 2002

Fomin Report Builder v.2.83

Fomin Report Builder v.2.83 is now available. The key feature of this release is support for the Taboga Barcode Library. There is now a new Live Update service available for registered FRB users. Look for the Live Update" button at the bottom of the left-side navigation bar on the web page. Using this instant service, you can always get download information and release notes regarding the latest product version. As well, there is a public support forum available through the web site. This is a primary news and support resource dedicated to Fomin Report Builder. Anybody can participate this forum as web visitor and/or subscribe to forum using e-mail address.

Posted Monday, April 29, 2002

Free ASCII Reports Template

Oleg Fomin has made a free ASCII Reports for CW template available. This template lets you create and print reports the same way you would when working with Clarion for DOS.

Posted Monday, April 29, 2002

New ClarioNET(tm) Deployment Manager

Michael Brooks has announced the forthcoming availability of the ClarioNET(tm) Server Deployment Manager and Load Balancer. This system provides complete management of ClarioNET deployments. A draft of the user documentation is now available for download. Pricing and availability will be announced within 10 days. The ClarioNET Server Deployment Manager provides these capabilities: a single control panel to manager your server deployment; a single point of contact for the client program to query available programs; automatically launches and close copies of the ClarioNET EXE Application Broker for each program instance. Only in-use broker EXE's are "live"; allows single or multiple program instances per EXE broker; automatic cleanup. Cleanup includes the following: checks that no temporary directories or files remaining; deletes orphaned temporary directories; deletes temporary may be left in the \PUBLIC directory; removes orphaned running EXE brokers; senses crashed program EXE files and terminates them; uses a winsock protocol to listen to incoming client requests so that always-on application broker instances and open ports are eliminated; provides detailed statistics for each running program; provides performance information for each running instance (this information is use by the Load Balancer module to launch new instances on the server that fits your load balancing scheme). For more details see the documentation.

Posted Monday, April 29, 2002

New Icons Available From Gitano Software

Jazz up your application with some good looking XP-style icons from Gitano Software. Each icon resource contains an XP, True Color, and 256 color versions to produce the highest impact in your application.

Posted Monday, April 29, 2002

Bug Poster Now Freeware!

The Bug Poster functionality is now freeware. This is an extension for your own applications which enable your customers to report bugs and enhancement-requests via email directly to you, the developer. You can then import this data easily into the "Buggy" bug tracking system.

Posted Monday, April 29, 2002

ZipFlash 2.3 Released

Sterling Software's ZipFlash template has been updated with the ability to look up and insert a County. Fields already handled are City, State, ZIP, Longitude, Latitude, Area Code & Time Zone. ZipFlash also includes Distance Calculation and map display (USA).

Posted Monday, April 29, 2002

etc-III Is Just Three Weeks Away

The East Tennessee Clarion Conference & Gathering will be held May 21-24 in Gatlinburg, TN. The ETC committee can no longer guarantee rooms at the Edgewater Hotel for Friday, May 24th. A few rooms are available next door at the River Terrace if your registration includes room reservations for Friday. There is still space available for registration to the conference.

Posted Monday, April 29, 2002

Imaging Template Update/New Imaging Product

NextAge Consulting has released version 1.13 of the Imaging Templates. The following changes are included with this update: 1. Added DeleteImageFromRegBrowse Extension 2. Added Embed before/after calling SaveAs 3. Added more choices to save the image 4. Added embeds before changing page 5. Changed the way thumbnails post events to the image to signal page change This allows programmer to use events between Event:User and Event:User+200 For their own purposes. 6. Added flag to template for the user to turn on/off confirmation messages for deleting current page/ all pages 7. Changed thumb nail caption and caption style to be prompt on template (can use variables if first char is !) 8. User can now select multiple files with the OpenButton control template 9. Added Export Button Control template that calls the filedialog and exports the image. 10. Added template prompts to let the programmer decide to show or not the warning messages when user clears single page or entire image 11. Give error when generating that certain template prompts are not filled in when required. Image file name prompts on the ImageControl Return Variable on Main Imaging function procedure 12. Added optional parameter to view procedure and call to viewing procedure to default to a particular page number 13. Added prompt to CurrentImagePath for user to specify starting directory for images.

Posted Monday, April 29, 2002

New ImageMan Templates Answers The XP Question

Unlike previous version of Windows, Windows XP does not ship with the Imaging

OCXs, therefore requiring you to purchase Imaging for Windows Pro from Eistream for each XP workstation that needs imaging functions. With NextAge's ImageMan templates, however, you can use the royalty-free ImageMan OCXs from Data-Techniques. For a limited time current registered users of the Imaging Templates can purchase the ImageMan templates for $100 ($49 off). Just choose the ImageMan Addon from the Checkout page and provide your Imaging Template Serial number. This offer is only valid for 30 days

Posted Monday, April 29, 2002

Icetips Magic Buttons

Icetips Software has released the Icetips Magic Buttons. These add background images to your buttons with a very easy, yet powerful extension template. You have complete control over what images are used, you can select images for individual buttons, exclude selected buttons or include only selected buttons. You can also specify variable image names and have the users pick the images they like. Magic Buttons comes with several backgrounds and four different themes for buttons. Each theme has three images: one for small toolbar buttons (25 by 25 pixels); one for regular buttons (25 by 75 pixels); and one for large buttons (75 by 75 pixels). The templates will work with any image that works natively with Clarion, i.e. bitmap, GIF, JPG or icons. Demo available. Compatible with Clarion 5 and Clarion 5.5, both ABC and Legacy. Buy it now for $49.00 and save US$30 during our introduction sale through April 30. On May 1 the price goes up to $79.00.

Posted Monday, April 29, 2002

RADventure Releases SQL Browse

RADVenture, the Clarion distributor in the Netherlands, has released an SQL browse procedure template, now available at ClarionShop.

Posted Monday, April 22, 2002

SoftMasters Toolbar Controls v1.0 Released

SoftMasters Toolbar Controls give your applications the same kinds of toolbars available in Outlook Express, Internet Explorer, Windows Explorer, Word, Excel and other applications. These toolbars are user-customizable. Viewlets-powered online demo available.

Posted Monday, April 22, 2002

Clarion Handy Tools Build O7A2.5

The Clarion Handy Tools Page has released its second, new build of 2002, available to registered subscribers. New features include: ; File and in-memory data compression; SQL browse extensions; More Dictionary Enhancement Features

(Including stuff for SQL tables); Browser Data Server Technology Now Does Triple Duty (Data-To-Browser, Data-To-Thin-Client, Standard Page Services); Thin-Client Technology Advanced (Includes Data Compression and SQL Query Pass-Through); Two new demo apps introduced (making 35 in all) to demonstrate the SQL extensions.

Posted Monday, April 22, 2002

EmailReport ABC & Legacy Templates 2.3

Release 2.3 of the EmailReport ABC & Legacy templates is now available for download. Now EmailReport works uniformly with all supported Report templates - Clarion, CPCS, RPM, DAS, and TIN. You can: Send reports in email body; Duplicate report in RTF format as attachment; Email reports as RTF attachment only; Save report to disc without emailing as RTF, DOC, or HTML. There is also a new generic embed which makes it possible to use EmailReport working with report templates other than those directly supported.

Posted Monday, April 22, 2002

VTIS ClarioNET Load Balancer At ClarionShop

The VTIS ClarioNET Load Balancer is now available from ClarionShop. You can also download a fully functional evaluation copy from this site. The demo does not expire, so is in effect is free for developers. The demo does display a registration message.

Posted Monday, April 22, 2002

INN Bio for 18-Apr-2002

This week's INN Bio features a guy who has been around the Clarion community for awhile. He lives in the same city as CIA Headquarters, but originally was from one of the less populated states in the U.S. In the past, he's done asphalt paving. Now, he presents us with a new idea for the browse-form paradigm, complete with screen shots.

Posted Monday, April 22, 2002

Pro Wizards News

The Icetips Professional Wizards are now in active beta.

Posted Monday, April 22, 2002

SysTree 1.0 Released

SysTree 1.0 is now available. This is a wrapper class for the tree-view common control, written in 100% pure Clarion code. It allows you to use tree-view controls, introduced by Microsoft with Windows 95, in your Clarion applications. This class

implements over 80 methods (17 of which are virtual event handlers), giving you full control over all the styles, settings and events without having to struggle around in the depths of API programming. The accompanying templates make the class easy to use. You can add a tree-view control to your application in less than a minute, without having to write a single line of code. Also included is SysDirTree, which is a tree-view control displaying file system folders. Demo available. SysTree can be purchased from solid.software or ClarionShop.

Posted Monday, April 22, 2002

G-REG Upgrade Amnesty

For two weeks only, all registered users will be able to upgrade to the current version of G-REGPlus for one upgrade price. If you haven't upgraded for a number of versions this is your chance to save money. Every user will be able to upgrade to the current version for the lowest upgrade price.

Posted Saturday, April 13, 2002

Insight Graphing Beta Back On Track

After a long pause last year, the Insight Graphing betas are back on track. It took a lot longer than expected to implement the planned drawing engine change, but the result has been worth the wait. The feedback from users has been overwhelmingly positive with the highlights so far being the speed increase, as well as the new functionality.

Posted Saturday, April 13, 2002

NetTalk Version 2 Going Gold On May 2, 2002

After a long period in beta, where the feature list changed considerably, NetTalk is finally going to "go gold". NetTalk version 2 (gold) will ship on May 2. CapeSoft is now in the final beta stages with new builds being released at a rate of about one per week. Current corrections and changes are pretty minor.

Posted Saturday, April 13, 2002

File Explorer Goes COM

CapeSoft has recently licensed the COM code produced by Andy Ireland of Plugware Solutions. Those who know Andy know that stability is his prime concern. So not surprisingly, by applying this new COM code to File Explorer, CapeSoft has achieved a massive leap forward in both speed and stability. Beta testers have reported 100% stability and speed increases of up to 200%. File Explorer currently costs $99.

Posted Saturday, April 13, 2002

HyperActive Now 100% ClarioNet Compatible

CapeSoft's HyperActive has now been made completely compatible with ClarioNET.

Posted Saturday, April 13, 2002

SecWin Improves SQL Support

CapeSoft's Secwin has radically improved SQL support and has been tested against Oracle, MsSql, MySql, Interbase, Pervasive and others. Only one feature (Concurrent login limitation / Who's currently logged on) has yet to be implemented in SQL releases.

Posted Saturday, April 13, 2002

CapeSoft Draw Update

CapeSoft's Draw has been undergoing some updates and bug fixes. Looks better, goes faster, is the mantra Sean is now chanting on a regular basis.

Posted Saturday, April 13, 2002

CapeSoft Safe Writer Released

During last year CapeSoft became aware of the fragility of the ZIP format as a distribution mechanism, and switched over to a new format, the so-called SAF, (pronounced "safe"). At the same time, CapeSoft released the free Safe Reader program. Now for $29 per user you can buy the Safe Writer program, which allows you to create your own SAF files. Because SAF uses 168 bit DES encryption it provides a level of protection from cracking which is not available with other systems. In order to unpack a SAF file you will need the free CapeSoft Safe Reader, which is freely distributable.

Posted Saturday, April 13, 2002

twTools 1.6 Released

twTools 1.6 is now available. This release includes the ability to send file attachments with email using twSMTP.

Posted Saturday, April 13, 2002

ETC NetTalk Session

At ETC III, on Tuesday 21st of May, from 2pm until 4pm Bruce Johnson will be running a special training session for NetTalk programmers. This will not be a sales session, but will be fore existing NetTalk users. There is no cost for this session, and all are welcome to attend.

Posted Saturday, April 13, 2002

New Versions Of gFileFind And PowerSearch

New versions of gFileFind and PowerSearch are available for download. This is a free update to all registered users. A new demo is also available.

Posted Saturday, April 13, 2002

EasyExcel 1.03 Released

New in EasyExcel 1.03: Fixed a bug in a Legacy Procedure Extension Template (didn't generate a EasyExcel Class declaration), so that you can use it now in a Source, Process and Report procedures; fixed a bug in a Save method, so it will clip the passed file name; added TrunkName parameter to AddSheet; SelectSheet now returns an errorcode if you try to select a non-existent sheet; ShowAlert and FileName parameters on Save are now omittable; OpenDoc.Init has a changed behavior; SetFormat has new Date and Time formats. There are also numerous new methods and code templates.

Posted Saturday, April 13, 2002

New KSbgc Template

The newly released KSbgc template allows your users to apply a custom background to application windows at run-time. This template adds a background selection submenu to any existing menu. Demo available; get the beta now and upgrade to gold for free.

Posted Saturday, April 13, 2002

ClarioNET Load Balancer Now Free For Developers

The ClarioNET Load Balancer is now available to developers at no charge. The fully functional evaluation version no longer expires. Ever. The only restriction is a nag message about registering, so if it is used in a commercial environment, then you will probable want to have your client register it. The new price on the Single-Server Multiple-Broker (SSMB) version will be US$250 per server. The new price on the Multiple-Server Multiple-Broker (MSMB) version for server farms will be available at US$650 per server farm.

Posted Saturday, April 13, 2002

CwEventLog 1.2 Released

Version 1.2 of CwEventLog is now available, and is a free update for all registered users. Changes: now comes as .LIB version, supporting local and standalone runtime libraries (this was required to work with ClarioNET - thanks to Tim McDonald for this contribution); documentation now is in HTML help format (.chm). CwEventLog is a small addon in form of a template and library that allows your Clarion applications to write to the Windows NT event log. Event logging in Microsoft Windows provides a standard, centralized way for applications (and the operating

system) to record important software and hardware events. It also supplies a standard user interface for viewing the logs and a programming interface for examining the logs. Event logging provides a means to merge events from various sources into a single informative story.

Posted Saturday, April 13, 2002

EasyVersion 1.02 Released

New in EasyVersion 1.02: bug fixes including redraw artifacts when switching tabs, global variables export, and external flag; variables are now ASTRINGs; there is a global variables initialization class; string constants can be automatically decrypted; DLL variable usage example included. This is a free update for all registered users.

Posted Saturday, April 13, 2002

Application Shell Monthly Special

During the month of April you can purchase the Office Plus shell for $495.00. That's a $300.00 savings off the regular price. Demo available. Office Plus is a complete, running set of business functions distributed in source form. This application is written using generic Clarion templates, so source code is similar from one module to another and modification is straightforward. Office Plus is network ready, fully integrated, and royalty free. This system includes over 250 procedures geared toward the smaller office. Application functions include: Payables; Invoicing and Receivables; Quoting and Job Cost; Contact Management; Inventory Control; Purchasing; Financials; Checkbook Reconciliation; Appointment Scheduling; Expense Report Tracking.

Posted Monday, April 08, 2002

SealSoft xPGP v1.0 Beta

SealSoft's xPGP is now in beta. This is a library, wrapper class, and template set for using PGP (Pretty Good Privacy) Shareware for Windows 95, Windows 98, Windows 2000 and Windows NT directly from a Clarion program. xPGP is created on a base of freeware Pascal source of the SPGP Library by Steven R. Heller. You must have the Windows version of PGP 5.5.x or 6.x installed on your machine for xPGP to work. PGP 2.6.x or any DOS version will not work at all. Features include:; Encrypt and Sign file, text with and without User Interface ; Decrypt and Verify file, text ; Select Recipients Dialog ; Fill Keys Queue and Get Key Properties ; Signing Passphrase, Key Passphrase Dialogs ; Confirmation Passphrase Dialog ; Check Passphrase Quality. Demo program available.

Posted Monday, April 08, 2002

Preview.IT Released

The preview.IT part of AnalyZe.IT lets your user re-run, preview, print or email all reports made with AnalyZe.IT.

Posted Monday, April 08, 2002

Door Prize For Your Next UG Meeting

User group presidents: Gitano Software would like to give you one copy of gFileFind to offer as a door prize for your next user group meeting. Also available: a free CD for each of your users (30 day advance notice required). The CD contains all of Gitano's utilities for trial plus some free stuff. CDs are offered to US-based groups only.

Posted Monday, April 08, 2002

Taboga Report Templates In Beta

Taboga Software has released a beta version of the Taboga Report Templates. Features include: Calculation/printing of grand totals; Efficient (range limits and more) printing of a detail band based on a child file; Hide a detail band so that it doesn't print; Hides a band (or bands) so they don't print; Clip a band's height so that it doesn't print taller than the defined height; Ability to write code break-dependant code (usable for example to simplify the calculation of group percentages, and other group/break related stuff); Print boxes around given detail/header/footer bands; adjusts (expands) the report's page length to account for changing size; Print a "memory" report (similar to what was available in Clarion 2.1 for DOS); Sort the report on any of the fields used on it, even calculated fields; Print a standard columnar clarion report in newspaper style. During the initial phase of the beta period, the templates will be available for only $79.00. Taboga Software reserves the right to change the price as the templates become more mature, or as the number of templates increase.

Posted Monday, April 08, 2002

BigOutlookTamer(tm) Control Template

BigOutlookTamer(tm) Control Template imports address book/contact information from Microsoft Outlook into a Clarion file such as a Customer, Vendor, or Contact file. Demo app available.

Posted Monday, April 08, 2002

PDF-Tools SDK Version 2 Released

Tracker Software has released version 2 of the PDF-Tools SDK. Included in this version are extended Clarion Template and Classes allowing Clarion developers to access virtually all the API Library using Clarion templates and methods. Also included are extended demo apps and updated documentation. Demo features

include: Merge single or multiple PDFs; Convert images to PDF (BMP, GIF, JPEG, PNG, TIFF, EMF/WMF) including direct conversion of Clarion Reports to PDF; Extract images from PDF (BMP, GIF, JPEG, PNG, TIFF); Extract pages from a PDF and create a new file. PDF-Tools has been tested with CPCS, FOMIN and RPM - demo apps for all are included. Existing users can upgrade from the password protected area.
Posted Monday, April 08, 2002

etc-III - 43 Days And Counting...

It's only 43 more days until the East Tennessee Clarion Conference & Gathering, featuring education from acknowledged leaders in our community, the tranquility of the Great Smoky Mountains National Park, and camaraderie with other developers and friends. What more could you ask for? Register today.
Posted Monday, April 08, 2002

INN Bio for 4-Apr-2002

The Icetips News Network is pleased to present its first bio from Finland! An expert in HVAC and SCADA (read the bio to find out what those are), he also loves "gizmos" and has made the coolest birdhouse. Lots of pictures, including the new gizmo, his beautiful family and country.
Posted Monday, April 08, 2002

twTools Version 1.5 Released

twTools version 1.5 has been released. In this version: twDataExport allows you to export data from a twInterfaceArray to a CSV file; twRuntimeFields lets you create runtime entry and prompt fields (this will be enhanced with more field types) on your windows, without using any variables or file fields. The example app shows how to use these new objects.
Posted Monday, April 08, 2002

0-HaZzle Price Increase

0-HaZzle will go gold next week, and the price will go up to $199 NOT $99 as stated earlier.
Posted Monday, April 08, 2002

RemFlash 2.0 Released

Sterling Software has released version 2.0 of RemFlash. This is a major update of the templates and has the following new features:Option to run an external EXE (or call one of your app's procedures) at the scheduled date/time; Ability to set reminder/messages recurrences - daily, weekly, monthly etc. Demo available.
Posted Monday, April 08, 2002

MySQL LIMIT Demo

Kristian Hyllestad has updated his MySQL demo which shows how to use LIMIT with MySQL/ODBC. You can get the installation password at clarion@spine.no.

Posted Monday, April 08, 2002

Clarion/MySQL Documentation

Kristian Hyllestad has updated his Clarion/MySQL documentation to include a browse procedure using ODBC API calls directly to build a Clarion queue.

Posted Monday, April 08, 2002

Taboga Barcode Library Update

The Taboga Barcode Library template now includes support for printing barcodes to page headers and footers, break headers and footers, and the report's form structure. Current customers who need this feature may simply download an updated library from our site. The same password applies.

Posted Tuesday, April 02, 2002

ExcelBond 1.2 Released

ExcelBond 1.2 is a template-only change and improves the way column headings are handled. By Default the field description entered in the dictionary will be used as the Excel column heading. If there is no description the field name will be used.

Posted Tuesday, April 02, 2002

C5.5 Gold To 5.507 Patch

SoftVelocity has released a cumulative patch that incorporates all updates that have been offered since the release of Clarion 5.5. It must be applied against the original 5.5 Gold install.

Posted Tuesday, April 02, 2002

EmailReport 2.2 Saves Reports To Disk

EmailReport templates by Vivid Help allow you to duplicate reports as RTF attachments or send reports in this format only. The 2.2 release lets you save reports on disk as well.

Posted Tuesday, April 02, 2002

BigTamer(tm) Templates & Utilities Add C4 Compatibility

The BigTamer(tm) Templates & Utilities now offer Clarion4 compatibility for most components.

Posted Tuesday, April 02, 2002

Gitano's gFileFind Shareware

Gitano Software's gFileFind is a flexible, customizable search utility. You can look for any file type in any drive and folder or a combination of drives and folders in your system and with a variety of filters. All of this is done fast. Additionally you can perform file operations such as copy, delete, move, edit and open. Edit and open operations require that a file association exists for that file type. You can also preview images fit to screen, actual size or tiled. All search options are saved automatically for reusability at a later time. gFileFind is being released at a price of $19.95 for a single installation. This is a stand alone, not for distribution utility. A sub version of gFileFind will be released today to allow developers to include gFileFind function in applications for distribution. Along with the function you will get the procedures in the demo (http://www.gitanosoftware.com/gffdemo.zip) to use as they are or modify to suit your needs.

Posted Tuesday, April 02, 2002

SealSoft Distributor Discount

The following products now are available to Clarion Distributors on ClarionShop with 40% discount: xWord Library; xWindow Settings; xSmartMacro; xDataBackup Manager Pro; xQuickFilter; xSearch; xDesignSuite #1; xDesignSuite #2; xProfessional Pack; xClock Pack; xSearch Pack; xXXL Pack.

Posted Tuesday, April 02, 2002

twMySQL 1.0 Released

Dan Pressnell's twMySQL allows your Clarion programs to connect to and use MySQL servers directly, without ODBC or Claron file drivers. Templates are provided that work with ABC and Legacy template chains, and do not change any of your existing templates. The browse template and object includes page loading functionality. The update template and object includes multiuser concurrency checking, and supports auto-incremented fields, whether done at the server or in your program. Hand coders who use MySQL should appreciate the flexibility in the kinds of SQL you can send to the server, and the results. Because you are not tied to a FILE or VIEW structure, you can do much more with SQL than you can with previous technologies. You can send any valid SQL command to the server without having any predefined FILEs at all. The code generated by the templates is very small, because the objects in twMySQL does the work we usually see templates doing. The twMySQL templates will co-exist with your current Legacy or ABC templates with no conflict.

Posted Tuesday, April 02, 2002

gCal 3.0 Update

A new update for gCal 3.0 is available for all registered users. All changes are having to do with the template, so if you own the code or the DLL version you will need to download this update. This fixes a refresh issue with the calendar control and legacy applications. The time picker control has been changed and it will require that you remove the control from your application and add it again. A "font bug" has been reported and verified with the control calendar and the "W" for Wednesday, the fix for this at present is to change the font.

Posted Tuesday, April 02, 2002

twTools 1.3 Released

Version 1.3 of twTools includes a twHTMLWriter, which you can use to create HTML pages or reports. You can also use it to compose email messages that you can send with the twSMTP object. It's handy for sending formatted invoices, etc., including tables. Demo available.

Posted Tuesday, April 02, 2002

GUI.IT Beta Released (Now 0-HaZzle)

Langaard Software has released a beta of the 0-HaZzle templates (former GUI.IT). You can join the beta program for 38$ at ClarionShop.

Posted Tuesday, April 02, 2002

EasyMultiTag 2.00 Released

EasyMultiTag 2.00 is now available. New in this release: support for legacy templates, and new batch processing methods. All known bugs have also been fixed.

Posted Tuesday, April 02, 2002

INN Bio for 28-Mar-2002

This week, the Icetips News Network is pleased to post a bio of one of the more prolific third-party providers. We mean he's prolific in providing products... although he and his wife have been prolific in having children, too (great picture of the kids!). Born in Spain, now living in Southern California, we think the Force is with him.

Posted Tuesday, April 02, 2002

RPM 5T for C55G Re-Released

By popular demand a newer install of RPM for C55G is now available for download. This install does NOT include long filenames except for source files. All DLLs, LIBs, PRJs and EXPs have been renamed and the templates updated to use the newer files. In the case of DLLs, they have also been recompiled.

Posted Tuesday, April 02, 2002

PostNet Barcodes For C55G Available

A recompile of PNet for C55G is now available for download. This is a free upgrade for all licensed users. If any AFE users are wondering, a new AFE install for C55G, as well as a newer 32Bit server installer, should be available shortly.

Posted Tuesday, April 02, 2002

Reader Comments

Add a comment

Detecting System Idle Time With Hooks (Part 2)

by John Gorter

Published 2002-04-17

As I explained last week, I have a need in one of my applications to detect the amount of time the entire system (not just my application) is idle. To detect system idle time I have to install the input hooks, `WM_KEYBOARD` and `WM_MOUSE`, because the `GetLastInput` function is not implemented in Windows 98. I could use the `WH_MOUSE_LL` and the `WH_KEYBOARD_LL` hooks since they call the hook in the thread that installed the hook, which means that when an input event occurs, Windows, just before executing the hook-function, will do a context switch to the thread that installed the hook, taking care that all globals are the only globals in the system, but these hooks aren't available in Windows 98.

This leaves me to implement my own system-wide globals using the `FileMapping` technique. I have to map a view of a kernel "memory file" object in my current address space and store the content of my variable in there. Other instances of my DLL will do the same, causing only one variable to become updated in the system after a hook function executes. You can install the appropriate hooks the following way in Clarion:

```
InstallHook procedure
hInst long
loc:dll cstring(20)
 code
 loc:dll = 'dll32.dll'
 hInst = GetModuleHandle (address(loc:dll))
 hKeyBoard = SetWindowsHookEx (WH_KEYBOARD, |
  address (MyKeyboardHook), hInst, 0)
 hMouse    = SetWindowsHookEx (WH_MOUSE, |
  address (MyMouseHook), hInst, 0)
 return true
```

The hook functions in this example are as follows:

```
MyKeyboardHook procedure (long nCode, |
  long wParam, long lParam) ! , pascal, long
 code
 AccessSystemVar (true)
 return CallNextHookEx (hKeyboard, nCode, wParam, lParam)

MyMouseHook  procedure (long nCode, |
  long wParam, long lParam) ! , pascal, long
 code
 AccessSystemVar (true)
 return CallNextHookEx (hMouse, nCode, wParam, lParam)
```

Note that the hook functions have similar prototypes; I could just use one function instead, but two different functions might come in handy when I want to use different behaviors on different events, or if I just want to provide "for future use," as Jim Kane would say.

Using just one function for more hooks might at first seem as a fast way to implement things but beware! The first parameter passed to the next installed hook is the handle to another hook procedure. Since I would have just one function for more types of installed hooks (WH_MOUSE and WH_KEYBOARD) I would have to implement logic that calls the correct next function. Whenever the originating event was a keyboard event I would have to pass the hKeyboard handle, and whenever it was a mouse event I would have to pass an hMouse handle as first parameter. This housekeeping can result in complex code!

The implementation of two hook functions was far easier than passing the correct hook based on a interpretation of the codes and parameters passed. Both functions call the function AccessSystemVar, which gets its name from the fact that the function accesses a system-wide counter variable. The function is implemented as follows:

```
AccessSystemVar procedure (byte par:updatevar)
 code
 loc:name = 'MyFileMapping'
 hFileMapp = CreateFileMappingA (|
   INVALID_HANDLE_VALUE, 0, PAGE_READWRITE, |
   0, 4, address(loc:name))
 dwCount &= (MapViewOfFile (|
    hFileMapp, FILE_MAP_ALL_ACCESS, 0, 0, 0))
 if par:updatevar then dwCount = GetTickCount() .
 if CloseHandle (hFileMapp).
 return dwCount
```

The AccessSystemVar function tries to create a FileMapping by name. This way

only the first instance actually creates a new `FileMapping` object while other instances receive a reference to it. This is the way to share kernel objects between processes. I can test whether or not the create function actually created the `FileMapping` by examining the `GetLastError()` return value, as I explained last week. If you want to read some more on this topic I advise you to read the book Programming Applications for Microsoft Windows by Jeffrey Richter (ISBN 1572319968).

After the `FileMapping` is located or created I can access the variable by mapping a view of the file to the process's address space with the function `MapViewOfFile`. This function isn't really necessary in Windows 98 since all `FileMappings` take place directly after the 2GB user address space, and directly writing to the address of the system variable should theoretically work, but doing so definitely results in a GPF on Windows2000! Using `MapViewOfFile` makes the code work on both Windows 2000 and Windows 98, so this is exactly what I did.

The `MapViewOfFile` function receives five parameters. The first parameter is the handle to the `FileMapping` object I've just retrieved, and the second parameter tells the function the kind of access desired. The other parameters tell the function the offset and the size of the memory to be mapped, in my case offset 0 and size 4.

When the memory is mapped to my address space I can increase the counter and close the handle to the `FileMapping` object. That's basically it. The reason I chose to close the handle is mostly to demonstrate a complete procedure flow. Keeping a variable with the handle returned from `CreateFileMapping` and closing it once for the application is definitely a more efficient solution. This is exactly what I did when designing the class mentioned below.

The remaining thing to do is write a little function that unhooks the installed hooks on demand, and another function that returns the current value of the system-wide variable. This latter function uses `theAccessSystemVar` function as follows:

```
GetIdleTime procedure
   code
   return AccessSystemVar (false)
```

The last function to implement is the code that removes the hooks from the chain:

```
RemoveHook procedure
   code
   if UnhookWindowsHookEx (hKeyboard) then .
   if UnhookWindowsHookEx (hMouse) then .
   return true
```

I realize that I should do some error checking every now and then. Let's say that I left it out for the reason of simplicity...

Where is the OOP

Since a lot of things in Clarion, and most other programming languages as well, are objects, I realized it would be handy to wrap this functionality in a little class. Then I could expand its functionality and gain the benefit everywhere I want to use system global variables. At its core this class only has to have an `Init` and a `Kill` method. The `Init` method creates a `FileMapping` object or gains access to an already created object, and the `Kill` method closes the handle of the FileMapping object. In the future I can implement arrays, queues and other types of info, but since that is not quite necessary now, I leave that as a practice to the reader. The definition of the class is as follows:

```
CFileMapping class, link ('CFileMapping.clw'), |
  module ('CFileMapping.clw'),    type
hFileMap       long
Init           Procedure (string szName, long lSize), long, proc
kill           Procedure (), byte, proc
             end
```

The only property the class needs is the handle to the `FileMapping` object. This way I can close the object later on in the `Kill` method.

The `Init` method takes two parameters. The first parameter is a string representing a name for the object. This naming is necessary to identify and access the `FileMapping` object from another process, as I explained in part 1. The second parameter is the size of the memory which needs to be allocated and mapped in the caller's address space. The `Init` method first creates the `FileMapping` object and, if successful, maps a view of it in the address space of the current process. It returns the address to the beginning of the memory just allocated. I can think of a situation were I want to share and access more variables. Gaining access to them could then be as simple as maintaining a queue of variables, their order and their size and then calculating the offset just before returning from `Init`. The code for both methods are quite trivial. The `Init` method looks like this:

```
CFileMapping.Init Procedure (string par:szName, |
  long par:lSize) ! , long, proc
loc:name cstring(20)
 code
 loc:name = par:szName
 self.hFileMap = CreateFileMappingA (|
```

```
   INVALID_HANDLE_VALUE, 0, PAGE_READWRITE, |
   0, par:lSize, address(loc:name))
 return (MapViewOfFile (self.hFileMap, |
   FILE_MAP_ALL_ACCESS, 0, 0, 0))
```

All the `Kill` method does is close the handle to the `FileMapping` object. The code looks like this:

```
CFileMapping.kill Procedure () !, byte, proc
   code
   return CloseHandle (self.hFileMap)
```

That's it! You should never again have to worry about simple system-wide variables (as long as you are developing for the Microsoft platforms).

Download the code

You can download the code to experiment with `FileMappings` at the end of this article. Run the executable and minimize the application to the taskbar. Watch the counter increase when there isn't input and decrease whenever something is typed; this will happen no matter which application which has input focus.

There are two versions of the code, one written in Clarion and one written in C++. If you experiment with the C++ version, I suggest adding a little code that shows the name of the EXE file in a message box when the DLL loads. You can do this by calling `GetModuleFileName()` with a first parameter of `NULL` in the `DllMain` function on the dwReason switch of `DLL_PROCESS_ATTACH`. Doing this shows you which EXE loads your DLL, and when. You can see that because of the system hook, each application does in fact load an instance of your DLL.

As you can imagine, there is a lot of other fun stuff I can do when I am in the process space of another process, including API Hooking, which I will discuss in a future article.

Download the Clarion source

Download the C source

John Gorter has been programming in Clarion for three years, before which he studied business informatics. He has just passed the MCSD exams and is now busy creating web applications with C55 Internet Connect. John lives in the Netherlands, and when not programming, reads about programming.

Reader Comments

Add a comment

I loaded dll32.dll and exe32.exe into the WinNT directory...
I've updated the Clarion source zip with a new version from...

Data Structures and Algorithms Part I: The Dynamic Stack

by Alison Neal

Published 2002-04-19

Like most Computer Science students moving into the Clarion environment, I found a distinct lack of knowledge among Clarion programmers regarding the use of data structures other than the Queue. On the newsgroups it happens time and time again: a Clarion newbie asks: "How do I implement a tree in Clarion?" or "How do I implement a list in Clarion?" These aren't questions about tree controls or list boxes; they're about established, standard data structures. And even Clarion developers with a good grounding in languages like C++ or Java often don't know how to create the Clarion equivalents.

I think these questions deserve an answer, if for no other reason than to bridge the knowledge gap between Clarion and other more mainstream languages. So, for all of those programmers in a similar position to mine, here is the first article in a series outlining how to implement standard data structures and algorithms in Clarion. For programmers who are not in this position, I hope some of these articles will provide insight into new ways of doing things.

The stack

What is a stack? It's a way of storing things in computer memory. Think of a stack of books on the floor. The first book you can take off the stack is always the last book you put on the stack. This is called Last-In-First-Out, or LIFO. Contrast this to a queue. I don't specifically mean a Clarion queue, but a queue in everyday life (or what most Americans call a "line"). Queues are easy to visualise: you queue up at McDonalds for your hamburgers, and at the super market register to pay for your groceries. The queue data structure (again, not specifically a Clarion queue) is where the first record accessed should always be the first one that was added. The

queue is a First-In-First-Out , or FIFO structure, and the stack is a LIFO structure.

The stack is a very useful tool, particularly for something such as an "Undo" command where it is imperative to store and retrieve the user's last action in the correct sequence.

Array implementation

There are two types of implementations of the stack: one is the array implementation, and the other is the dynamic implementation. The array implementation is precisely that, it uses an array as part of the data structure. Here's a Clarion declaration:

```
TheStack    CLASS,TYPE
stop           USHORT
nodeVal        ULONG,DIM[10]
            END
```

The `stop` variable maintains a count of how many items have been added to the `nodeVal` array (the stack), and thus provides an easy way of indexing the top value when required.

I've wrapped this declaration in a `CLASS` structure purely to make it possible to use it inside any class. It could just as easily be declared as a `GROUP`, but `GROUP` references are not legal inside `CLASS` declarations. Basically, a `CLASS` with no methods is identical to a `GROUP`. And by using the `TYPE` attribute I can have more than one identical stack declared at a time without having to write duplicate structure declarations. .

The array implementation, however, is limited by the size or dimensions of the array, unless you use a dynamic array. In Clarion you can achieve this by using strings (a.k.a. Binary Arrays). You "grow" the array by creating a new string of a larger size. You then copy the data from the old string into the new string and dispose of the old string. Where there is a lot of data being manipulated this process can carry a large overhead, which often counterbalances any speed gained by manipulating contiguous memory. The Dynamic array is however considered as an independent data structure; I may discuss it in a later article.

Dynamic implementation

When you don't know the maximum number of nodes, but you suspect that this number could get quite high, it is better to use a dynamic implementation. This is

the declaration of a dynamic stack:

```
stackNode       CLASS,TYPE
prevNode          &stackNode
nodeVal           ULONG
                END

s               &stackNode
t               &stackNode
```

Again, I've wrapped the stack declaration in a CLASS structure. Instead of an array of node values, there will be one stackNode instance for each node. The stackNode structure contains a ULONG for the value, and a reference variable to point to the previous stackNode. The stack is built like a chain, where the first node is the floor on which the stack of books is to be built.

One critical point is that I need to know where the floor is. Why? Well, if I don't know where the floor is I cannot tell if there are any books on the stack and I cannot tell where to stop processing.

When I start to use the stack, the first book (node) is pushed on to the top of the floor. The first book (node) includes a NULL reference to prevNode which indicates where the floor is. Then more books (nodes) are pushed onto the stack. With every node containing a reference to where the previous book is stored in memory. These references to the previous top of the stack are the links in the chain.

This type of data structure, where the declaration incorporates a reference to an object of its own type, is called a *recursive* structure. As I mentioned before, the only way to do this inside a Clarion class is by declaring the structure as a typed class (or, if you prefer, a typed QUEUE). How useful is it to build an ever-growing stack of books? Think about the "Undo" command. . Each command is pushed onto the stack, and then when the undo command is called, the last command is popped (removed) off of the top of the stack. The command stored prior to this then becomes the top of the stack. I'm not really interested in anything else but the top of the stack - that's all I'm ever looking at. . The other nodes are kept just in case I need to go back more than one step.

When a node is popped off the top of the stack there has to be something physically stored for it to be successfully removed. It's impossible to remove a book from a stack if there are no books and therefore there is fundamentally no stack. So I must check first to see that I'm not looking at the floor.

Sometimes it's also nice to know what's on the top of the stack. Say, for example, a

user wants to know what the last action was but doesn't necessarily want to undo it. Well, in that case I would just return the value of the top node.

Included in the downloadable source at the end of the article is a demo application and a small class that implements a stack. The methods required for the stack class are as follows:

Method	**Description**
`InitStack`	Initialise the stack
`Push`	Push a node on top of the stack
`Pop`	Pop a node off of the stack
`Top`	Find out what's on top
`Empty`	Check to see whether the stack is empty
`OutofMemory`	Check to see there's still memory
`KillStack`	Cleanup time

This class has only one property, a reference to a stacknode:

```
s &stackNode
```

The initialization of the stack is very simple.

```
 S &= NULL
```

This simply identifies the floor. In processing where the top node equals `NULL`, I know I've either reached the end of the chain or there is nothing in the stack. To push a node on to the stack I do the following:

```
   t &= s
   s &= NEW(stackNode)
?  ASSERT(~s &= NULL)
   s.nodeVal = YourVal
   s.prevNode &= t
```

There are several things happening here. First a reference is taken of the current top node of the stack for later use. When the first book is pushed on to the stack this reference will equal `NULL`. Then memory is allocated for a new node. The `ASSERT` statement checks to see that the memory is available and that it has been

successfully allocated. If it isn't successful then the new node s will equal null. (The question mark on the left of the ASSERT statement tells the compiler not to include this assert check if debugging is switched off at compile time.) Then the new node is assigned its value and is linked by reference to what was the original top node.

If s = NULL to start and the new value is 1, the stack would look like this: 1 -> NULL. If another node were then pushed onto the stack with the value of 2, it would look like this: 2 -> 1 -> NULL. Thus the stack always terminates with NULL, which is the floor on which the stack sits.

Note that no memory was allocated for t; this temporary reference variable is merely made to refer to exactly the same memory space as s was occupying originally.

Here's the code to pop (remove/delete) a node on the stack:

```
   t &= s
?  ASSERT(~t &= NULL)
   s &= t.prevNode
   temp = t.nodeVal
   DISPOSE(t)
   RETURN temp
```

Here t refers to the same memory as s, and a check is performed to see that t is not the last node. If t is NULL, the program can't continue because the next assignment won't work. This assignment moves the previous node into the top spot, and then disposes of what was the top spot.

If the stack is currently 2 -> 1 -> NULL, t is made to refer to exactly the same memory as s, and s then becomes 1 -> NULL. The node 2 -> is then disposed of, so that there are no memory leaks. This doesn't effect s.

To find out what is on top of the stack is easy:

```
?  ASSERT(~s &= NULL)
   RETURN s.nodeVal
```

As long as there is something allocated to s, as long as the stack is not empty, you can see what is in the top node.

As you may have guessed by now it is all-important to know whether the stack is empty or not. There are two ways of doing this: one is the ASSERT check, and the other is a NULL test, as follows:

```
RETURN CHOOSE(s &= NULL,TRUE,FALSE)
```

This NULL test basically says: "Am I looking at the floor? If I am then return TRUE, and if I'm not then return FALSE. You can then wrap this code in an isempty() function, and rewrite the pop and top function ASSERT statements to:

```
? ASSERT(~isempty(s))
```

Remember that before you call the top and pop functions you should also check to see that the stack isn't empty:

```
 IF ~s.isempty() THEN thisVal = s.pop().
```

Another useful function is to check whether there's any memory left to continue. The "out of memory" function can be written the same way as the empty function, only you are passing it a specific node to check, so the push function becomes this:

```
t &= s
s &= NEW(stackNode)
IF OutOfMemory(s) THEN RETURN FALSE.
s.nodeVal = YourVal
s.prevNode &= t
RETURN TRUE
```

Since Clarion allocates heap memory for NEWed objects, it's highly unlikely you'll get an out of memory error unless your stack is immense, or your available memory is severely constrained.

As the stack is a recursive structure it also opens the doorway for recursive functions. For example if I wanted to add up all the nodes in the stack I could do this with the following code:

```
StackSum        Procedure (), LONG
T  &stackNode

   CODE
   T &= S
   RETURN Stacksum(t)


StackSum        Procedure (stacknode t), LONG

   CODE
   IF NodeVal = NULL THEN RETURN 0.
   RETURN T.NodeVal + StackSum(t.prevNode)
```

When killing any data structure that's been allocated dynamically you have to be

very careful that there are no memory leaks in the code. Wherever memory is allocated it has to be disposed. Here is the code for the `kill` function:

```
LOOP WHILE ~isempty()
   Pop()
END
```

The `kill` function simply calls the rewritten pop function until there is nothing left to dispose.

Summary

The stack can be applied to and built in any application where it is imperative to maintain data in sequence. Often this use is at a low level: compilers widely use stacks to keep track of function calls, for instance. But stacks are useful anywhere you need to keep track of something that might need to be undone or returned to at a later stage.

The downloadable zip below includes a class implementation of the stack. There is also a very simple application to show the class in use.

Download the source

Alison Neal has been using Clarion since 2000, whilst working for Asset Information Systems (AIS) in Auckland, New Zealand. Some years ago (at the tender age of 19) Alison graduated from the Central Institute of Technology in Wellington, New Zealand with a major in Cobol. She also has a BA in English literature and has studied Computer Science, Philosophy and Information Systems. AIS is an independent division of Asset Forestry Ltd, and has a team of five programmers developing almost exclusively in Clarion. AIS also offers web (ClarioNET) and email services for the customer who needs everything. The company has many and varied customers bridging across a wide range of industries including Telecommunications, Forestry & Agriculture, Manufacturers, Military & Government, Legal & Financial, and Retail.

Reader Comments

Add a comment

Alison - I think you have indeed hit the nail on the head...
Alison: Very good article and example. I could see...
Alison - Great job. Will you also be showing how to perform...
Definitly worthwhile things to know. What about...
Alison, Congratulations on a fantastic, descriptive,...

Clarion Magazine

It's an XML World!

by Jim Kane

Published 2002-04-25

I've never seen anything adapted as widely and quickly as XML has been by the computer community. In fact, every one of the consulting projects I've taken on in the last year have involved XML in one form or another. When yet another offer to do an XML project came in about a month or two ago, I figured why not? Should be simple, I thought! Famous last words...

After digging into the project a bit, I realized the project required setting data types for all the XML elements; for this project the data types involved were `STRING`, `DATETIME`, and `BINARY` (base64). Fortunately, use of the MS XML DOM parser was mandated for this project, and it handles all those things with ease. Unfortunately the Clarion OLE Control does not work well with this component because it returns long strings which are often truncated by Clarion. The MS OLE control also uses some data types such as safe arrays which the Clarion OLE control can not handle.

Luckily, Clarion also provides Interfaces which make calling the DOM object's interfaces quite easy. You can find the interface prototypes in the accompanying code in the `xmldomcl.inc` file. For an example of prototyping and using interfaces, I refer you to one of my previous article series.

The MS XML parser is a COM object that reads and writes XML data. It contains two XML parsers. One is called SAX - this is the one I wrote about previously. The other is commonly called the DOM (Document Object Model) parser. For this project the DOM was a requirement. Although at first the wide array of interfaces in this object is a bit daunting, once you get Microsoft's terminology and view of an XML file down, the parser is actually quite pleasant to work with.

To get started with the DOM parser, the first step is to initialize COM, create an instance of the DOM object and obtain a pointer to the IxmlDomDocument2 object. In the accompanying download there is a class called xmldomcl. The Init method in that class does this through the use of my StdCom class:

```
SELF.StdCom.initcom(pDebugmode) !init com
  !init the dom
  lpInterface=SELF.StdCom.GetInterface(address(clsid:Dom3)|
     , Address(IID:IxmlDomDocument2))
  if ~lpInterface then
    SELF.ErrStr='IE5 or higher must be installed to continue'
    return return:fatal
  end
  SELF.IXmlDomDocument&=(lpInterface)
```

As noted in the code above IE5 is required for the MS XML Dom.

The general format of the XML I needed to create was this:

```
<?XML version="1.0"?>
<Root xmlns:dt="urn:schemas-microsoft-com:datatypes">
  <DocName dt:dt="string">Sales Data</DocName>
</Root>
```

The first line is called a processing instruction. The IXMLDOMDocument interface just obtained has a method called CreateProcessingInstruction, which takes as input the text seen in the processing instruction, as bstrings. It returns an IXMLDOMNode interface that represents the processing instruction node. The code to do this is in the CreateNewDomDocument method, and with error checking code stripped it looks like this:

```
SELF.strcl.CwtoBstrAlloc('XML', lpBstr1)
SELF.Strcl.cwtobstralloc('version=''1.0''',lpbstr2)
IXMLDomDocument.CreateProcessingInstruction(lpbstr1,lpbstr2,lpNode1)
```

At this point I have an isolated node interface, but the node is not in the XML document or tree. To put the node I just created into the tree, I use the IXmlDomDocument class's InsertBefore method.

For the first insertion I insert before NULL. The NULL needs to be passed as a variant, which is a 16 byte group that contains a data type and the actual data. The data type for NULL is vt_null, which is just an equate that happens to have a value of 1. So I set the type field to 1 and the value or actual data field to 0. That represents a NULL.

The next twist is that the variant is a 16 byte group that has to be passed by value. In other words, all 16 bytes (or four LONGs) must go onto the stack. To do that I use the Clarion built-in memcpy function to copy four bytes at a time from the variant structure to a LONG; I repeat that four times. Once the variant has been copied, those four LONGs are passed as parameters. The code to convert the variant to four LONGs is located in the StdCom class in the variantbyValue method. If you use that method, passing a NULL to the insertbefore method looks like this:

```
!clear the variant group and
! set the data value to 0
variantinit(address(var))
!set the data type
var.vartype=vt_NULL
!convert to four longs
SELF.StdCom.VariantbyValue(address(var),v1,v2,v3,v4)
SELF.IXmlDomDocument.InsertBefore(
 lpNode1, v1, v2, v3, v4, lpNode2)
```

This inserts Node1 before NULL, or the processing instruction at the start of the document. Node2 is a return value which isn't needed, and represents the node just inserted. Now both node1 and node2 can be freed by calling their release methods. All this is done in the xmldomcl.CreateNewDomDocument method. At this point, the XML document is just one node long - the processing instruction <?XML version="1.0"?>

Next I want to add a root node, such that after this next step the document will look like this:

```
<?XML version="1.0"?>
<Root></Root>
```

To create the root element, in the IXMLDomDocument method I use the CreateElement method. Elements and nodes are very similar; I'm not entirely clear on what the difference is. I like to use elements rather than Nodes at times because the Element interface has better methods for adding attributes, which I'll get to next.

To create the Root element, I call CreateElement in a manner very similar to the usage of createprocessingelement. After the root node or element is created I add it to the tree with a special property called DocumentElement. The code to do this is in xmldomcl.CreateRootNode and looks like this:

```
SELF.strcl.cwtobstralloc('Root',lpbstr1)
SELF.IXmlDomdocument.CreateElement(lpbstr1,lpRootElement)
```

```
SELF.RootElement&=(lpRootElement)
SELF.IXmlDomDocument.put_DocumentElement(lpRootElement)
```

Notice I'm saving the IXmlDomElement interface in a member variable called RootElement, because I will be using it to append other nodes as I build the XML tree.

Next I want to add the attributes to the Root node. After the attributes are added the document will look like this:

```
<?XML version="1.0"?>
<Root xmlns:dt="urn:schemas-microsoft-com:datatypes"></Root>
```

This particular attribute defines a namespace. The namespace tells an XML parser that the data types defined by Microsoft are going to be used, and when a data type from the Microsoft definitions is used it will be prefixed with dt:. For example, if the data associated with the DocName node is a STRING, and the data associated with the CreateDate is a DATETIME, you would use the following:

```
<?XML version="1.0"?>
<Root xmlns:dt="urn:schemas-microsoft-com:datatypes">
  <DocName dt:dt="string">Sales Data</DocName>
  <CreateDate dt:dt="datetime">2002-03-20T20:40:15</CreateDate>
</Root>
```

The code to add the xmlns attribute to the root node in xmldomcl.CreateRootNode is this:

```
SELF.strcl.cwtobstralloc('xmlns:dt',lpbstr1)
variantinit(address(var))
var.vartype=vt_bstr
SELF.strcl.cwtobstralloc('urn:schemas-microsoft-com:datatypes',var.VarValue)
SELF.StdCom.variantbyvalue(address(var),v1,v2,v3,v4)
SELF.RootElement.SetAttribute(lpbstr1,v1,v2,v3,v4)
```

At this point the basic framework of the XML document is built, and I can begin adding child nodes. Notice how the DocName and CreateDate nodes are between <Root> and </Root>. They are considered child nodes of Root. Likewise the string data 'Sales Data' is between <DocName> and </DocName>, so it is a child of (but not on the same level in the tree as) DocName. To add DocName I create the DocName element as before; to put the element into the tree I use the AppendChild method on the parent element (SELF.RootElement).

Once the node is added to the tree I can use SetAttribute, or take a short cut to add the child text node containing 'Sales Data'. I could repeat the sequence of

creating a child node and appending it to the <DocName> parent, but there is a quicker way. The IXmlDomElement interface has a special property called datatype that, given the intended datatype such as string, adds the proper attribute. Once I have set the data type property, I can call NodeTypeValue and create the node and append it in one step. The combined code which adds the line <DocName dt:dt="string">Sales Data</DocName> is contained in the xmldomcl.AddChildNode and xmldomcl.AddTextData methods; with error checking stripped the code looks like this:

```
!create the node called NewElement
SELF.StrCl.CWToBstrAlloc('DocName',lpbstrNodeName)
SELF.IXmlDomDocument.CreateElement(lpbstrNodeName,lpNewElement)
NewElement&=(lpNewElement)
!append the new node to the root element
SELF.RootElement.AppendChild(lpNewElement)
!Set the datatype
SELF.StrCl.CwToBstrAlloc('String',lpbstrdatatype)
NewElement.Put_Datatype(lpbstrDataType)
!add the text node
variantInit(address(var))
var.vartype=vt_bstr
SELF.strcl.cwtobstralloc('Sales Data',var.varvalue)
SELF.StdCom.variantbyvalue(address(var),v1,v2,v3,v4)
newelement.put_NodeTypedValue(v1,v2,v3,v4)
```

Although all the BString conversions, building variants, and error checking adds a lot of "noise" to the code, overall, using data types ends up simplifying the code a great deal. Also, once a data type is declared the parser checks that any data added with NodeTypedValue conforms to the required data type. For example, in the AddDateTimeData method, I use a format string to put the Clarion datetime into the correct format:

```
DATETIMESTR = FORMAT(THEDATE,@d010-) |
  & 'T'& FORMAT(THETIME,@T04)
```

If the picture is changed, from d010 to perhaps d4 the method will then fail because the parser will see the data is not in the correct format.

Reading back the data is also accelerated by using data types. Getting the NodeTypedValue property returns the data for a node in one step in the xmldomcl.GetTextData method. Compare that code with the code in the XmlDomcl.GetUnTypedData, which uses the traditional approach of getting a list of child nodes. Once the list of child nodes is returned, the programmer needs to loop through the list and find a text node, and then finally read the text property to get the text.

Binary data is treated the same as any other data type, with one change. The data needs to be presented to the NodeTypedValue property in the form of a safe array. Fortunately I have a handy safe array class (as previously described) that can convert binary data in a string into a safe array, with one line of code.

For this project, I used the BYTES( ) function to get the file size and used NEW( ) to allocate a string the size of the file. Next I use either the DOS driver or the API equivalent to read the file into the sting. Although the data is stored in a sting, no string function such as clip or len() is ever used on the data. String is just an convenient Clarion data type to store binary data in. Once the string loaded with binary data is ready, I create the XML element called BinaryData (returned as lpElement) and add the bin.base64 data type to the node like this:

```
XmlDomCl.AddChildNode(|
Address(SELF.RootElement),| !parent node to append to
'BinaryHexData',|  !node name
'bin.base64',|  !data type
lpElement) !returned pointer to the new element/node
```

Next the binary data stored in the string 'binarydata' is added:

```
xmldomcl.AddbinaryData(lpElement, bindata,size(bindata))
```

The code inside AddBinaryData looks like this:

```
NewElement &IXmlDomElementtype
Code
!convert the binary data in the string to a safearray
SELF.sarraycl.StringToSafeArray(psa, bindata, size(bindata))
!Put the safe array (psa) into a variant group
variantinit(address(var))
Var.varvalue=psa
 var.vartype=vt_Array+vt_UI1 !arrays of bytes
!convert the variant to four longs
SELF.StdCom.variantbyvalue(address(var),v1,v2,v3,v4)
!put the data into the passed element
NewElement&=(lpElement)
NewElement.put_NodeTypedValue(v1,v2,v3,v4)
```

At this point the XML document looks like this:

```
<?XML version="1.0"?>
<Root xmlns:dt="urn:schemas-microsoft-com:datatypes">
  <DocName dt:dt="string">Sales Data</DocName>
  <CreateDate dt:dt="datetime">2002-03-20T20:40:15</CreateDate>
  <BinaryData dt:dt="bin.base64">MTIzNDU2Nzg5MA==</BinaryData>
</Root>
```

Summary

The `xmldomcl` class accompanying this article contains a few other goodies demonstrated in the sample program. The `doNodes` method calls the `TakeNode` method for each node. Inside `TakeNode`, the data for the node as well as the attributes (datatype) can be read. The sample program displays the information in a little message box. There also are `loadfromdisk` and `savetodisk` methods to load and save XML to and from disk. The `Kill` method is also needed to clean up and uninitialize COM.

Although the complete project required a few other things not covered here, like schemas, validation, and sending the XML via http, the same DOM component can do those things too, and it's free! While this article only scratches the surface of what the DOM can do, it's clearly a powerful tool for manipulating XML data, and for billing clients!

Download the source

Jim Kane was not born any where near a log cabin. In fact he was born in New York City. After attending college at New York University, he went on to dental school at Harvard University. Troubled by vast numbers of unpaid bills, he accepted a U.S. Air Force Scholarship for dental school, and after graduating served in the US Air Force. He is now retired from the Air Force and writing software for ProDoc Inc., developer of legal document automation systems. In his spare time, he runs a computer consulting service, Productive Software Solutions. He is married to the former Jane Callahan of Cando, North Dakota. Jim and Jane have two children, Thomas and Amy.

Reader Comments

Add a comment

Required Fields Times Two

by Carl Barnes

Published 2002-04-26

A lot has been written lately in Clarion Magazine about validating required fields. At the risk of going over some well-worn ground, I think there are a few points that haven't yet been clearly made. Here's my approach to required field handling.

A required field is a field the user must not leave blank, if a string, or zero, if numeric. Clarion provides an IDE checkbox to mark a field as required, but to insure that the REQ (required) attribute is enforced requires a second step by the developer. And to provide a friendly user interface requires a little more work.

I'll admit it: I used to just check the required box and assume that Clarion would do the rest. How's it work? Don't know, don't care. It must work. I told Clarion that field was required, and in Bruce Barrington's ideal language that should be enough. I didn't even need to test it. But then I noticed that at times, required fields were not being enforced. So from then on I checked the required checkbox and hoped it would work. But hoping wasn't very effective, so I read Richard Taylor's Fabulous Manuals and did some testing. I learned that there are two steps every developer must take to get a required field and have it enforced.

Step 1: Make the field required

The first step in implementing a required field is obviously to make the field required. In the Clarion language this is done by adding the REQ attribute to the window control, e.g. ENTRY(@s20),USE(Emp:Name),REQ. Or you can dynamically set the required attribute at runtime using PROP:REQ. But you don't have to do this in code - it's far easier to use the window formatter and checking the Required box on the field properties Extras tab, as shown in Figure 1.

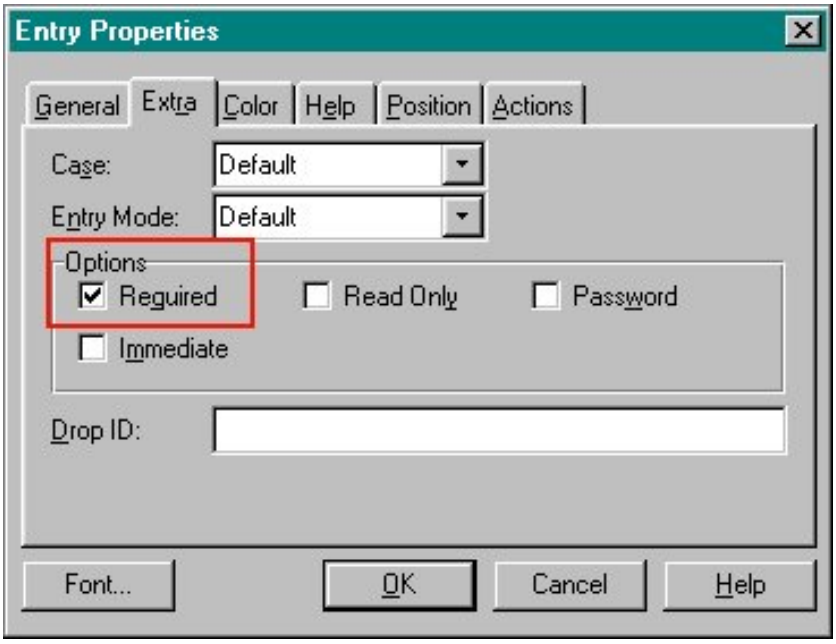

Figure 1. Setting a field's REQ attribute

It's possible that this checkbox is already set for you. The typical way you end up with a required field is when defining a field in the dictionary. In the Field Properties on the Validity Checks tab you select the "Cannot be Zero or Blank" radio button. Then when the field is populated in the window formatter (either by a wizard/wizatron, or by you, the developer) the Required box will be checked automatically. Because there can be a long time period between designing the dictionary and designing the window, it's easy to forget which fields will get tagged as required. Figure 2 shows a screen shot of a required field in the dictionary.

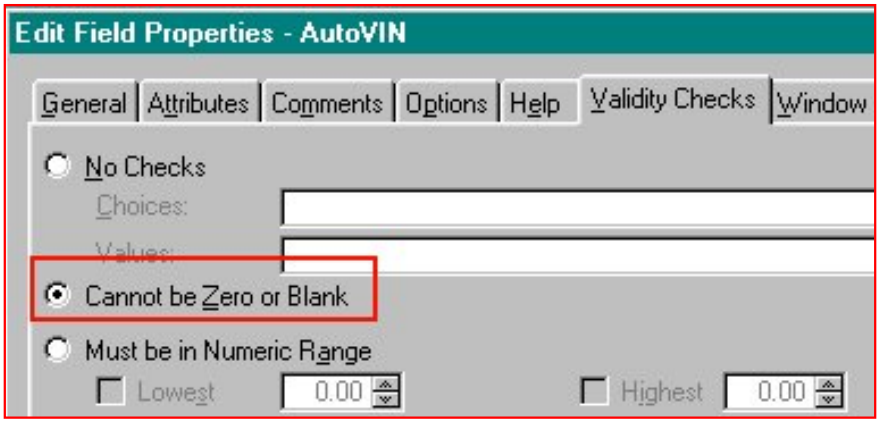

Figure 2. The dictionary attribute that results in the REQ attribute

Step 2: Enforce the required field

The important second step you must take is to implement or turn on the enforcement of the required fields. Failure to take this second step allows required

fields to be left empty by the user.

You have two basic choices: turn on automatic runtime library (RTL) handling, or implement your own manual handling. You have to understand both because automatic handling can prevent manual handling from working at all. It does this by simply not sending you an `EVENT:Accepted`, which you would normally use to test the validity of an entry.

It is very important to know that the default Form template (and Save button extension template) handling of required fields is automatic. Every other procedure template (Window, Browse, Process, etc) does not have any type of required field enforcement. Marking a field as required does nothing in a Window template (unless you've also populated the Save button extension, which effectively turns a Window into a Form).

Using Window templates can get you into trouble. Most required fields I implement are on Forms, and Forms enforce required fields (i.e. the "most part" of required fields that seem to work most of the time). But then I create a quick Window template-based procedure to gather the parameters for a report. If I want to use required fields on a Window template I must implement the enforcement, or those fields can be left empty.

Automatic required field enforcement

The first thing to know about automatic enforcement is that it has a very poor user interface; in fact it has no user interface. It simply selects the first empty field and might beep. To the user it can appear that the program is frozen, or that the Ok button does not work. I recommend you turn automatic handling off and use manual enforcement. But I'm getting ahead of myself.

Automatic required field enforcement is very easy to implement, and as noted above is the default behavior in Forms. Essentially you get it for free. By remembering to check one box you can have automatic enforcement on any window.

There are two ways required fields can be automatically enforced by the RTL: by the use of a button with an `REQ` attribute, and by `SELECT()` AcceptAll Mode.

Required button automatic enforcement

A button can have the required attribute (I call these REQ-Buttons) in exactly the

same way a field does. You either add the REQ attribute to the button control code e.g. BUTTON('Ok'),USE(?Ok),REQ, or in the window formatter button properties you check the required box on the extras tab. But a required attribute on a button has a different purpose than on a field. A required button is not a button the user is required to press.

When an REQ-Button is pressed, the RTL first checks all required fields. If a required field is found empty, the RTL selects it and sounds a beep. The button does not get an EVENT:Accepted if there are empty required fields. That is very important to know. An REQ-Button is mutually exclusive with all remaining options. You cannot use other types of enforcement unless you uncheck the required box on the button. In the Form template the default Ok button has the required attribute, so you'll need to remove the REQ to use the techniques discussed below.

The REQ-Button is a great and easy enforcement method because it does insure there are no empty required fields, and all it requires is checking the required box on the Ok button. It is somewhat of a problem that you do have to remember to do it for non-Forms. You can fix the default window definitions used by the IDE by editing the Defaults.CLW file in the LibSrc directory. Edit each Button('Ok') and add the REQ attribute. You may have to make these changes each time you install a new version of Clarion. As well, these changes will only affect new Window template procedures that you create after you've made the changes. If you populate your own Ok button you'll still have to remember to check the Required checkbox.

Accept-All SELECT() automatic enforcement

The second method of automatic enforcement is performing a SELECT(), i.e. a SELECT statement with no parameters, to initiate Accept-All mode. The main feature of Accept-All is sending an EVENT:Accepted to each control. A lesser-known feature of this mode is that the RTL first checks whether each control is required. If a required control is empty the Accept-All mode is ended and the user is left with the empty control selected. The control does not get an EVENT:Accepted. Again there is no user notification that a required field is empty, not even a beep.

One of the features of the Form template's Ok button (the Save button template) is it performs a SELECT(). But a Form also has the required attribute on the Ok button by default. As noted above the REQ-Button RTL behavior will check for empty fields first and prevent the button's EVENT:Accepted. So the Accept-All will never see an empty required field.

Required button manual enforcement

Manual enforcement of required fields is easy using the INCOMPLETE() function, which returns the field equate of the first empty required field. If all required fields are filled, INCOMPLETE() returns zero. I'll remind you again that you must remove the REQ attribute from the Ok button or your manual code will never see an empty required field. The Accept-All will not interfere with the manual code and actually provides a backup to the manual method.

With the small amount of code shown below placed in the Ok button's EVENT:Accepted embed (before generated code) you can ensure all required fields have been entered. The INCOMPLETE() function only identifies the field, so you must add a SELECT(INCOMPLETE()) to move the focus to the empty field. You also need to issue a CYCLE to prevent further accept loop processing. A simple MESSAGE() provides a minimal but effective user interface. The user will not think the program has hung, and will in fact have a clue as to what needs fixing.

```
IF INCOMPLETE()
   SELECT(INCOMPLETE())
   Message('The selected field must be entered.', |
       'Required Field',Icon:Asterisk)
   CYCLE
END
```

Since INCOMPLETE() returns a field equate you can enhance this code to identify the problem field by name to the user:

```
IF INCOMPLETE()
   SELECT(INCOMPLETE())
   CASE INCOMPLETE()
   OF ?Emp:Name ; ReqField = 'Employee Name'
   OF ?Emp:SSN  ; ReqField = 'SSN'
   ELSE         ; ReqField = 'selected field'
   Message('The '& CLIP(ReqField) &' must be entered.', |
       'Required Field',Icon:Asterisk)
   CYCLE
END
```

Finding the selected empty field on a busy screen can be difficult. That damn caret seems to keep getting smaller and harder find as my eyes age. An enhancement that will make the empty field easy to spot is to color it yellow. For some entry types, like an Option , this can look ugly. But if a user is trying sneak out without giving me the data I am legally entitled to, things just might have to get ugly! To allow for undoing the color, declare two LONGs to track the prior incomplete field and its color.

```
IF IncomFld   !Was there a Prior Incomplete?
   IncomFld{PROP:FillColor}= IncomFldSaveColor
END
IncomFld = INCOMPLETE()
IF IncomFld
   SELECT(IncomFld)
   IncomFldSaveColor = IncomFld{PROP:FillColor}
   IncomFld{PROP:FillColor}=Color:Yellow
   Message('The selected field must be entered.', |
       'Required Field',Icon:Asterisk)
   CYCLE
END
```

(One other presbyopian tip I have will help you in the Clarion Editor. When your compile has errors, you can press the Goto Error button and the IDE will position you to the line with the error. The trouble I have again is finding that little caret on the big screen. The trick I use is to hold down the Shift key and press Home. That will select all the text on the error line (making it white on black) and make it very easy to see. If that does nothing then the caret is in column one and Shift+End will select the line.)

A template solution

If you are going to use a manual method like the one above to handle incomplete fields, the best way to implement it would be with a template. I would design it as an Extension template that was very smart and required little developer input. The ideal template could be added to a default Form template (with REQ on the Ok button) The template could automatically find all REQ-Buttons, remove the REQ attribute and insert code for a better user interface. This would make an interesting Clarion Challenge, particularly as I don't think there has been a template challenge.

Clarion is like an onion

Clarion is like an onion. It makes you cry? No! Clarion has layers. Or maybe I should say Clarion is like parfait. Parfait has layers and everybody loves parfait. The layers of Clarion are: Embed Code, Template Generated Code, ABC Code, Language, and Runtime Library. When trying to understand exactly how Clarion handles something, e.g. required field processing, it usually is best to start at the bottom and understand exactly how the language elements and RTL work.

In this article those language elements are BUTTON, ENTRY, REQ, SELECT() and INCOMPLETE(). The best place to learn about these is reading the help and LRM. Sometimes you have to dig around a little since not all the topics are logically

linked, but I find the Clarion documentation to be very good overall.

If you really want to understand how Clarion handles a particular behavior, your next step should be writing an all-source procedure, i.e. Clarion language only. You should not use the templates since that will add another layer and possibly confuse things. To this end I have included a small source project I created for this article. Once you are have a solid understanding of how the language works it becomes easy to understand the template and ABC code impacting the process.

Required field logic overview

The below pseudo code tries to summarize what could happen on a window when the Ok button is pressed. This window has required fields. With the structure I have tried to convey what behavior prevents other behavior. For instance, you can see that REQ-Button checks come before EVENT:Accepted. Also this code covers all the possibilities; to have all this code running in a form would introduce some redundancies and some unreachable code.

```
User Presses Ok
POST(Event:Selected,?Ok)  (Selected always gets sent)
IF Ok{PROP:Req}=True (RTL checks Button for Required)
      RTL Checks for Required fields Empty first
      IF Empty Found
         Select Empty Field
         Beep
         Cycle - Done
      End IF
END
POST(Event:Accepted,?Ok)
OF Event:Accepted
   IF INCOMPLETE()
      Manual Code to
        Inform User
        Select( Incomplete() )
        Cycle - Done
   End IF
   SELECT() - Start AcceptAll Mode
      LOOP Thru all Fields
          IF Field{PROP:Req}=True (Field is required)
             IF Empty
                Select(Field)
                End AcceptAll
             EndIF
          End If
          POST(Event:Accepted,?Field)
      END
      POST(EVEMT:Completed)
End Event:Accepted
```

Summary

Required field checking requires more than just marking the entry field as required. You must ensure that required checking is enforced or required fields can be left empty. Now that you understand how it all works you can make the choice between "Tastes Great" (manual method with a nice user interface) and "Less Filling" (automatic RTL method).

Only the Form template and Save Button template have automatic required checking enabled. None of the other procedure templates enforce required fields. I recommend that as you are designing windows you turn off the RTL required field enforcement on the Form and handle all required field enforcement the same way using the `INCOMPLETE()` function. If you want to go the "Less Filling" route, then be sure your Ok button has the `REQ` attribute.

Download the source

Carl Barnes is an independent consultant working in the Chicago area. He has been using Clarion since 1990, is a member of Team TopSpeed and a TopSpeed Certified Support Professional. He is the author of the Clarion utilities CW Assistant and Clarion Source Search.

Reader Comments

Add a comment

The main points in this article are certainly important...
This covers the subject well. Another simple approach...
Parfaits are good but onions are better. They are more...

SoftVelocity's Upcoming Product Technology - Abstract

Published 2002-04-29

Clarion Magazine's Reviews Editor, Tom Hebenstreit, recently attended SoftVelocity's Technology Workshop in Florida, and came away stoked about the company's soon-to-be-released product and its future direction. He's written up his experiences for Clarion Magazine in considerable detail. Remarkably, almost every technology Tom covers in these articles was actually demonstrated by SoftVelocity at the workshop. Some of these products, of course, still need a bit of finishing work. Highlights include the following.

- The Legacy templates are back! There are no guarantees that they will get every ABC feature (or vice versa), but they will again be actively supported.

- The next release of Clarion will be 5.6, which will include a new threading model. Clarion is finally being brought into the modern age with true preemptive operating system threads. True threading may not necessarily change an out-of-the-box Clarion application, but it means powerful new capabilities for the language and for your applications.

- SoftVelocity will now support ADO (ActiveX Data Objects), Microsoft's preferred method of accessing data sources in Windows. Attendees were treated to a look at Pierre Tremblay's 'ADO Explorer,' and a 'Query Center' that let you drag and drop column values from your browse to another list box where it would use both the column name and the selected value to build a WHERE clause.

- Clarion/ASP is a new product that generates ASP-based web applications. ASP (Active Server Pages) is a server-side scripting environment developed by Microsoft for generating dynamic web pages, and it is included for free along with every installation of Microsoft's IIS web server. There are numerous new templates that provide the Clarion/ASP capability, offering a tremendous amount of control over the resulting pages.

- Clarion/ASP will run on Windows only because it uses ADO, but for fans of Linux and other non-Windows operating systems there will be a similar PHP product. The concepts, templates and prompts will be identical to the ASP product with the exception of how you specify the database connections.

- Clarion users will also have XML support in the box, featuring four different ways of utilizing XML. For the most part, these will consist of new classes and/or templates that will wrap around existing XML technologies developed by Microsoft and others.

- Seminar attendees were also given some insight a number of possible future products. Unlike the rest of the seminar presentations, these future possibilities are strictly in the "maybe" stage, but they offer a tantalizing glimpse into SoftVelocity's future plans.

In all, the technology workshop was a huge success, with attendance at the workshop and the associated Essentials and Mastery classes exceeding all expectations. Look for more workshops (and more coverage) from Clarion Magazine in the future.

Because of the length of the report, it's available as two documents, Part 1 and Part 2.

Reader Comments

Add a comment

SoftVelocity's Upcoming Product Technology (Part 1)

by Tom Hebenstreit

Published 2002-04-29

I just attended the SoftVelocity Technology Workshop in Florida last week, and if I had to choose one word to describe it, that word would be: "WOW". They packed a huge amount of information into the two and a half day conference, and let me tell you, I almost injured myself trying to write fast enough to take all these notes (never mind trying to read them later).

In any case, some of the points I'm going to talk about were official presentations, others were sidebars, and some came to light in the question and answers periods. One really amazing thing about the Workshop, though, was that just about every technology I'm going to mention was actually demonstrated. There was no sales hype and no smoke and mirrors, just real programs doing real things. Granted, there is still work to do to make some of them ready for public consumption (for example, wrapping new classes in templates) but from the first minute of the first session, the emphasis was on the real world.

There's a *lot* to cover, so here we go (in no particular order)...

'Legacy' is Legacy

In moving to the ABC templates, TopSpeed basically decreed the death of the original Clarion procedural templates. No more work was to be done on them, no new features would be added and they branded those templates with dreaded 'legacy' moniker seemingly in the hopes that they would just go away (fat chance, guys!). In a refreshing change, SoftVelocity said that those templates will once again be simply known as the Clarion templates, and that they will start adding new features again (see the R&D section below for possible improvements). No

guarantees were made that the Clarion templates would be given every ABC feature (or vice versa!), but at least those templates are back in play and will no longer be treated as an ugly stepchild.

So... Clarion template chain users, rejoice!

One more Clarion 5.5 release

There will be one more Clarion 5.5 release, primarily to support the imminent release of ClarioNet v1.2. Some changes were made to the RTL (the Run-Time Library) to address a few ClarionNet issues, and thus a final 5.5 patch has to be released. This also means that if you intend to upgrade to ClarionNet 1.2, use of this new 5.5 release is required.

On Future Clarion Releases (5.6 vs 6.0)

Clarion 5.6 will be the next major release (maybe two to three months from now), and as you'll see, it will really have enough new technology in it to qualify as a larger upgrade than an incremental dot release (certainly it has more core changes and new features than the jump from Clarion 5 to 5.5). So why isn't it being designated as Clarion 6? Well, TopSpeed promised the world again and again that Clarion 6 would have a shiny new 32-bit IDE, and 5.6 will not have that. (Slight editorial digression: Personally, I could care less -- the Clarion IDE doesn't sell my programs, the features I build into them do. In my humble opinion, SoftVelocity is concentrating on the right things at this moment in time, i.e., stability and adding support for new core technologies that let me create and sell competitive products.)

So, yes, the next version will have a '6' in it, but it's just on the wrong side of the dot. And before you ask, SoftVelocity is working on 6 right along with the new stuff for 5.6. It will have a new IDE (they are exploring a number of options), but beyond that, they are keeping their cards close to the vest.

As to pricing, the 5.6 upgrade is slated cost no more than the upgrade from 5 to 5.5 did. They haven't actually set the price yet, but they have set that as the ceiling in their discussions. Me, I just want it all right now -- put it out, guys, and let me give you some money!

Core 5.6 Change: Real Threads

Real threads, you say? I thought Clarion already had threads. What is that START( ) thing all about then?

The answer to that is that all versions of Clarion (including 5.5) actually have what you could call psuedo-threads, with the appearance of threading being done by compiler sleight-of-hand behind the scenes. In terms of true operating system threads, all Clarion programs only run on one thread, regardless of how many times you call the START() function. Every time you switch 'threads' in your program, what actually happens is that the RTL does a switch for you - saving off the context of the current 'thread', then restoring the next one so that it can continue on the one actual thread. This round robin swapping happens continually as your program executes. With the psuedo-threads, there is only one actual memory location for threaded buffers and variables, with the contents being swapped in and out at each switch.

The new threading model gets rid of all of that - Clarion is finally being brought into the modern age with true preemptive operating system threads. When you say START(), you will now be starting another real operating system thread complete with its own private set of buffers, variables and so forth.

So what does this mean for you? To quote Bob Zaunere's presentation, "It changes everything, and it changes nothing."

By this, he meant that the way they implemented the new threading model should not break your existing programs. For the most part, we wouldn't even know anything had changed if they hadn't told us. So why do it? Here are a few of the more obvious benefits:

- More than one thread can be active at a time, and execute its ACCEPT loop for its top window. Instead of appearing to do more than one thing at a time, your programs can actually do them.
- A thread can execute a handler for registered events, i.e., you are not required to have an ACCEPT loop to process events for a thread.
- You can SUSPEND and RESUME threads, using two new language functions.
- You can have a true IDLE procedure for the main thread.

For those more inclined to low-level tinkering, here are a few other considerations:

- Threaded classes are fully supported.

- The new RTL and file drivers support preemptive thread switching.
- The EXTERNAL and THREAD attributes are no longer mutually exclusive.
- The Clarion RTL has per-process and per-thread initialization flags in the header, so every process gets its own copy of the RTL. This means it can now be considered re-entrant and thus Clarion can be used with many low level Windows functions that up to now were considered out of bounds because Clarion was not 'thread safe'.
- Clarion programs can use INTERFACEs for access to critical sections and mutexes for thread synchronization. (If you have to ask what those are, I haven't a clue - it seems to make the bit-twiddlers very happy, though.)

With any change of this magnitude, there are a few possible situations where prior Clarion code may break, e.g., having the THREAD attribute set properly is much more important than it was before.

One area that required special attention from SoftVelocity (and many changes to the RTL and ABC internals) was in the initialization of most FILE-related classes. For example, where you could previously rely on the result of a reference assignment with a threaded buffer being the same across all threads (remember, there was only one actual copy of any structure), each true thread now has its own private copy, and thus a different address for each buffer. To help ease this part of the transition, SV says they will document their internal ABC changes (which they have already made) as a teaching tool to help people convert their own classes. Note that conversion is only needed if your classes relied on the same type of single-instance reference assignments.

I'm sure we will all be hearing a lot more about the new threading model in the coming months as more details come out and 5.6 goes into beta release. To assist SV in flushing out the impact of these changes as early as possible, the company is considering offering a free service during their alpha testing where they will let you submit your application and they'll try compiling it with the new 5.6 threading code. I have no idea how they'll handle the logistics of such a program, but the dedication it shows to supporting their customers (i.e., us) is fantastic.

> (Editor's note: For a more detailed treatment of Clarion's threading model, as well as some educated guesses about what to expect in the 5.6 threading model, see Dave Harms' article on threading.)

5.6 Feature: File Driver Triggers

File driver triggers would give you the opportunity to have callbacks from the file drivers before and after certain operations (reading a record, writing a record, etc.) It would let you create the equivalent of SQL triggers while still using ISAM files (TPS, for example.) Very powerful, and could also be a real boon for developing systems that are targeted at both ISAM and SQL installations.

ADO Support

ADO (ActiveX Data Objects) is Microsoft's preferred method of accessing data sources in Windows, especially Access and SQL Server, and SoftVelocity is now going to be actively supporting it.

Some people will be pointing out here that it is already possible to use ADO with Clarion. Well, sure - you can use ADO right now in your programs if you don't mind getting your hands dirty defining all of the required interfaces to the ADO COM objects and handling all of the low level interaction yourself. It is, however, definitely not for the faint of heart (or even the basically lazy, like me), and there are currently a few pieces of the puzzle that are still lacking in Clarion 5.5. Clarion Magazine has published a number of articles detailing how to do ADO in 5.5 - they are very instructional as to what you won't have to do in the future.

Some key points about the new ADO support:

- ADO will not be replacing the existing Clarion file driver technologies. Each method has its strengths and will continue to be developed.
- ADO is oriented around SQL. It is possible to use it for ISAM files (heck, you can use it to read Excel files), but it would be akin to using an elephant gun on a gnat.
- At its heart, it will be new Clarion classes wrapped around the standard Windows ADO COM objects that are provided by Microsoft.
- Direct support for the Variant data type used by ADO/COM is being added to the language and RTL, and it will take care of the conversion to native Clarion data types (one of those missing puzzle pieces I mentioned).
- It will be released first as ABC extensions.
- SV also plans on eventually creating an ADO based template set that is not ABC based.
- Eventually, you will be able to create the destination structures (group or queue) on

the fly based on whatever is returned in the resulting RecordSet

- Execution of stored procedures and parameterized queries are supported.

Two primary objects will handle the interaction with ADO: a Connection object and the aforementioned RecordSet object. The first handles the basic interaction with ADO (connecting, setting transaction levels and so forth), while the second one handles your record level transactions (sending queries, specifying the cursor and lock types, looping through/dealing with the resulting record set, etc.)

Since a RecordSet is an amorphous kind of object to deal with, the ADO equivalent of a file buffer will be a GROUP or QUEUE. A TableMapper object will greatly ease the process of mapping your RecordSet columns to your group/queue fields, matching the column names with a single call to the mapper. Only columns and fields that have matching names will be mapped, so care will need to taken in this regard.

Once that is done, you call the mapper again to swap the values to and from the Clarion structure and record that the ADO cursor is pointing to. This needs to be done each time the ADO cursor position is changed. A simple code example might look like this:

```
Hr = MyRecordSet.Open(SQLQuery, MyConnObj, CursorType, |
       LockType, Options)
If Hr = S_OK
   ! Set up the initial mapping for the result set
   Hr = MyRecordSet.MoveFirst()
   MyCustMapper.MapRsToGroup(MyRecordSet,MyGroup)
End
```

One program that ADO presenter Pierre Tremblay showed was a utility he created called 'ADO Explorer'. Originally built to help him debug the ADO classes, it has turned into a very useful tool for interactive ADO testing. You can setup your connection, cursor, locking and multiple other parameters, testing it as you go. Once you have the connection, you can use it to pass SQL SELECT, WHERE and ORDER BY clauses and then browse the resulting RecordSet. In a really neat trick, the tool tip for each button shows the ADO code that will be sent back to the database engine. While the user interface was still raw at the workshop, it really brought home the flexibility of using ADO, since it worked equally well for any data source you pointed it at (no file drivers and thus no error 47). The plan is for ADO Explorer to be polished up and included with the final ADO classes.

One point to keep in mind when considering using ADO versus a native Clarion file

driver: by using ADO you make yourself much more dependent upon the Windows setup of each machine you install your program on. For example, since ADO is a part of the Microsoft Data Access Components (MDAC), MDAC must be installed on the target machine before your program will work. You are also more dependent on having the right versions of those components installed, and are more vulnerable to changes Microsoft may make to those components.

ADO/SQL Templates

After Pierre kicked off the workshop by presenting the inner workings of the new ADO support (talk about being tossed directly into the deep end of the pool!), Mike Gould had the much easier task of demonstrating new templates that will make using ADO as simple as any other data access method in Clarion. Even though work on those templates was at a very early stage (two to three weeks), they already had a number of very impressive features, such as:

- A Connection Builder on the global Connection Manager template that lets you easily select and set up your ADO connection, then test it right there. It will then copy the resulting validated connect string into the template for use by your program.
- An ADO browse template that had features such as column resizing, hiding and sorting (ascending and descending) via clicking on the list box column headers.
- Additionally, you could have multi-level browse sorts on the fly. For example, click on State to sort by state, then click on City (while holding down the Ctrl key) and it would add City as the second component of the sort order.
- This one had to be seen to be believed: A 'Query Center' that let you drag and drop column values from your browse to another list box where it would use both the column name and the selected value to build a `WHERE` clause. For example, if you clicked on the `State` column of your browse, and the selected record had `'CA'` as the value, dragging it up to the query list and dropping it would parse it and create a `WHERE` clause for State equal to `'CA'`. Comparison operators, values and connectors for clauses (e.g., `AND/OR`, etc.) were all available, allowing the creation of complex queries.

All this was done using the regular IDE to set up the List box format, specify the underlying SQL statements, drop on templates and so forth. To take real advantage of the ADO templates, though, it will still be very helpful to have some knowledge of SQL syntax since the templates let you explicitly provide the SQL clauses used by the browse.

Other planned features will include an ADO-based update form, report and a process template for looping through non-related rows (otherwise you would just update multiple records in one pass with an SQL UPDATE statement.)

Read Part 2

A longtime Clarion user, Tom Hebenstreit is an admitted tool junkie who refuses to go straight and code without his arsenal of third party products. During those rare moments when he isn't either using or writing about Clarion, he indulges his twin passions for blues and beer by performing around Southern California in a variety of totally-obscure-but-famous-any-day-now rock and blues bands.

Reader Comments

Add a comment

Re: Mutexs For the best reading on all types of Kernel...
Carl, Mutexes are used to syncronize threads in the same...
Another good book on threading is "Multithreading...
Thanks for the clarifications, guys. I figured *someone*...
"On the Fly..." Tom, When discussing ADO, you stated...
I've added a link in the text to my article of last fall on...
And yet another text. 'Foundations of Multithreaded,...

SoftVelocity's Upcoming Product Technology (Part 2)

by Tom Hebenstreit

Published 2002-04-29

Haven't read Part 1 yet? Click here.

Clarion/ASP Templates (New Product)

Clarion/ASP is a new product that generates ASP-based web applications. ASP (Active Server Pages) is a server-side scripting environment developed by Microsoft for generating dynamic web pages, and it is included for free along with every installation of Microsoft's IIS web server. The basic idea of ASP goes something like this:

- You embed program code (either VBScript, JavaScript or any other supported scripting language) into web pages, and give the page a special extension such as '.asp'.
- When an ASP page is requested from the web server, it first hands the page to another process that 'runs' the page, parsing and executing the commands embedded in the ASP page. These commands can include instructions for accessing a database, validating information or just about anything else. The output from this process is the 'real' web page, i.e., one that contains the pure HTML that your browser will display.
- The resulting HTML page is then given back to the web server, which sends it to your browser. The key concept is that your browser never sees the ASP scripts, only the results their execution by the web server.

How does this product fit in with the other Clarion web technologies such as Internet Connect, Web Builder and ClarioNet? Quite nicely, thank you.

- Internet Connect runs in a web browser, requires Java and uses the Application broker to run the applications.

- Web Builder also runs in a browser and requires the Application Broker, but doesn't need Java on the client side.

- ClarioNet is a thin client that doesn't run in a browser, but also uses the Application Broker. You run an actual Windows program on the client PC to establish the connection and communicate with the broker.

The common thread between each of the above is the reliance on the SoftVelocity Application Broker to either run the programs or to mediate between the programs and a web server. Using the broker gives the applications some great capabilities, but it also complicates things in that it adds issues of scalability (how many people can access the broker on a single machine before it is overwhelmed) and distribution (finding an ISP that will let you install it on their machines).

The web pages and code generated by Clarion/ASP, on the other hand, have no such restrictions. They are standard web pages, use standard Microsoft technologies and can be hosted on any ISP that supports ASP (virtually all of them). Since no broker is involved, scalability is limited only by the hardware is it run on, and thus the ASP code will work equally well for small projects up through high-volume public web sites. Does this negate the value of the other three technologies? Certainly not - all four approaches have their strengths. With Clarion/ASP you simply get another tool for your toolbox.

Some of the more interesting features include:

- As is typical with Clarion, your Data Dictionary drives the process. Formatting, prompts, lookups, must be in list, etc., are all used in the creation of the ASP pages. Note that it is even more important for Clarion/ASP, as it doesn't use the Window formatters in creating your pages.

- All the files required for a Clarion/ASP web application can be generated (HTML, ASP and cascading style sheets) using template extensions you add to a Clarion application.

- Clarion/ASP can be added to existing applications, with no impact whatsoever on your Clarion code. ASP/HTML pages are generated only for those browses and forms that have the Clarion/ASP extensions, so you can selectively create web pages from a subset of a larger Windows application.

- All Clarion/ASP data access is accomplished via ADO, and thus is oriented towards

using a SQL engine as your data source. This is not to say that you can't use Access files, etc., but it is not recommended.

- The user interface has been separated from the business logic via the use of 'processor' pages. Processor pages include the ASP scripting, data access and logic, with the results of the scripts being merged with HTML 'template' pages to produce the final page (think of it like a glorified mail-merge). This allows a web designer to modify the HTML interface as needed without affecting the underlying business logic of your application.

- Generation of the HTML and style sheets (i.e., the user interface) portions of the pages can be turned off either globally or on a per procedure basis so that regenerating the processor pages won't overwrite interface changes your web designer made.

- JavaScript is used on the client side (i.e., the web browser) to handle validation, lookups, popup help windows and even an optional popup calendar for date fields.

- All Clarion/ASP messages can be customized, and the templates will come with support for English, Spanish and French right in the box. Selecting the language is as easy as making a choice on a template and regenerating.

- Clarion/ASP has a built-in security model for logging in users and validating access (99 levels).

- Emphasis was placed on being able to integrate the pages generated by the Clarion/ASP templates with existing web sites. SoftVelocity understands that the world doesn't start and end with them, and that their products are more likely to be used in enhancing existing sites than creating them from scratch (although you can certainly do that as well).

John Iacovelli had the unenviable task of going through all of the available template prompts for Clarion/ASP right after lunch, and believe me there are a *lot* of them. This is a good thing, though, as you are given a tremendous amount of control over the resulting pages. Even better, 99% of the prompts are primarily for overriding default behavior. A fully functional Clarion/ASP site can be created by wizarding an application from your dictionary, adding the Clarion/ASP extensions globally and to each browse and form you want to create pages for, specifying the global database connection info (using the same interactive Connection Manager mentioned in the section on ADO) and then clicking on generate all.

I could go on and on about the Clarion/ASP feature set (I only skimmed the surface), but let me just sum up by saying this is one impressive tool. Look for more details in future Clarion Magazine articles.

The cost for the Clarion/ASP product is projected to be US $595, and SoftVelocity is hoping to release it within two to four weeks of the end of the conference. Check out this link for all you need to know about ASP.

One question that was asked was if the pages generated by the ASP templates would run under Linux, using an ASP emulator like Sun Chili!Soft ASP. Basically, the answer is no. Even though the ASP scripting portion would probably run, all of the data access code is done through ADO, and that is not available on Linux. If Linux is your goal, though, all is not lost...

Clarion/PHP Templates (New Product)

The PHP Templates will be another add-on product for Clarion, only instead of generating ASP/ADO code they will generate code based on the open-source PHP scripting standard. Basically, the concepts, templates and prompts will be identical to the ASP product with the exception of how you specify the database connections. The real advantage of these templates is that you can host the PHP pages they generate on any operating system and web server combination that supports PHP, and that means Linux, Unix or, if you are feeling perverse, even Windows.

One drawback to scripting standards such as ASP and PHP is that data access is not intrinsic to the language itself; it is provided by vendor- or community-supplied extensions (ADO, in the case of ASP). This means that with PHP the data access code can be different for various back ends, and that makes the generation of data access code much more complicated (imagine if the Clarion templates had to generate completely different code for every Clarion database driver.) In the case of PHP, though, open source has come to the rescue again. SoftVelocity found an extension called 'ADOdb', a class library that wraps around various PHP database extensions and provides an ADO-like generic interface to a large number of databases. As you can expect, this not only allows SoftVelocity to generate generic code, it should let them reuse a lot of the work already done in generating ASP/ADO code.

An interesting side note is that the ASP and PHP templates are compatible with each other. You could add both template sets to the same app and generate ASP and PHP pages at the same time. Oh, yeah, and Clarion code, too!

The cost for the PHP Templates should be about the same as the ASP add-on, but that has not been set in stone yet. Availability is projected to be within a month or

two of the release of the Clarion/ASP product.

And what does PHP actually stand for, you ask? According to some sites, it was an acronym from the very earliest version of the program ('Personal Home Page Tools'). Officially, it now seems to be one of those recursive name-contains-the-name things that Unix types love so much: 'PHP: Hypertext Preprocessor'.

For more information on PHP, visit any of the following sites:

- http://www.php.net
- http://www.zend.com
- http://php.weblogs.com/ADODB

XML, XML, XML

XML (eXtensible Markup Language) is one of the hottest new technologies to hit the scene in a while (a search today for 'XML' on Google counted 11.5 million hits). It is the basis for the latest buzzword ('Web Services'), is projected to become THE method of transferring information between systems and also is the language spoken by rising stars such as SOAP (Simple Object Access Protocol). Virtually all major SQL engines either already have built-in XML support or it is rapidly being added. Browsers such as Internet Explorer can understand and display XML files.

And now Clarion users will also have XML support in the box (whoopee!). Even better there will be not one, not two, not three, but four, count 'em, four different ways of utilizing it (SoftVelocity has stated that they regard XML support as a core technology for the future.) For the most part, they will consist of new classes and/or templates that will wrap around existing XML technologies developed by Microsoft and others.

A full explanation of XML is way beyond the scope of this article, but suffice it to say that there are a LOT of XML resources available on the web (do a search right here on Clarion Magazine and you'll get 15 or more hits, including recent articles by Jim Kane on using some of the XML parsers mentioned below in Clarion 5.5.) I've also listed some important links at the end of this section. A few basic concepts are helpful, though:

- An XML document is an ASCII file containing structured information delineated by

'tags' that look very similar in format to HTML tags.

- Information without context is fairly useless, so XML also provides for a way to describe the structure of an XML document so that the recipient can understand it. This can take the form of a Document Type Definition (an older method) or an XML Schema (a newer, more powerful, and more flexible method that will become the standard).

- An XML document can be invalid (basic structure is bad, e.g., missing or mismatched tags), well formed (structure is ok, but no guarantee about the actual data) or validated (structure is ok and it passes all validation rules in the associated document schema).

So, basic XML support requires being able to read, parse (and optionally validate) incoming XML documents (data), as well as generating valid XML documents (and optionally schemas) from your own data. Here's a brief (and simplistic) description of each of the ways reading and writing XML will be supported in Clarion:

- Using DOM (Document Object Model) via CenterPoint's open-source XML class library. DOM loads an entire XML document into memory at one time. This makes it very good for manipulating, displaying and doing mass transformations on a document, but not so good for very large files. If you view an XML document in Internet Explorer, you are using DOM (that's why you can view it as a tree, collapse and expand nodes, etc.)

- Microsoft SAX Parser. This parser is event driven and processes an XML document sequentially one element at a time. This makes it very good for dealing with large files, or when you just want to process a subset of a file. SAX is implemented as a COM object, by the way.

- ADO/XML. If the backend has built-in support for XML (SQL Server 2000, for example), ADO can be used to read, process and write XML files directly from or into the database (including generating an in-line schema) via SQL statements (see the `'FOR XML'` extension to the `SELECT` statement, for example.)

- `PropertyTree` class. Hmmm - this is in the Powerpoint slides of the workshop book, but I don't recall it being discussed (either that, or my brain was just on overload.) More details will be forthcoming, I'm sure.

All in all, it is very encouraging that Clarion users will be able to simply say "I can do that" when asked about utilizing a 'hot' new technology like XML.

- http://www.ucc.ie/xml/ (a good FAQ introduction to XML)

- http://www.w3c.org/xml (the World Wide Web Consortium XML home)
- http://www.cpointc.com/html/home-en.html
- Microsft MSDN XML home page (lots of good links)

Future Goodies (SV R&D projects)

SV said that at any given time they have lots of little Research & Development projects going on, and in this session they talked about a few of the current ones.

Now, before I say anything else here, I promised Bob Brooker (the presenter) that I would make something very clear: Anything listed in this section may or may not actually appear in future products, and if it does appear, it may not even remotely resemble what I describe here. They are just giving us an inside glimpse of things they are thinking about, experimenting with or generally tossing about. So, with that said, let's proceed (good enough, Bob?).

Note: Where templates are involved, I've specified the target chain as well.

- Form Record Navigation (Both ABC and Clarion). This lets you browse your file via a form instead of a list box. Along with the standard next record/previous record type buttons, Bob also showed the ability to specify the sort order and setting ranges and filters on the fly from the form (way cool). The navigation controls can be on the form, or on the application frame.
- Browse Extender/Edit-In-Place (Clarion templates). He demonstrated EIP in the Clarion templates, with support for Entry, Spin, Drop, Combo and Ellipsis lookup buttons. You could also right-click on the list box header to auto fit (resize) columns.
- List Formatter (Clarion templates). This would be an extension template that would let you create custom listbox layouts at runtime (hide/restore columns, etc.). Changes could be saved and restored.
- Enhance Selected Control (Both). This is a global template that modifies the appearance of the selected control (e.g., the control with focus on a form). Options include changing the background color, bolding the text, setting an alternate color for required fields and having an optional visual indicator. The default indicator is the '»' character, or you can specify your own.
- Business graphing (Both). These would be control templates that would take advantage of the built-in Clarion graphics primitives to create 2D and 3D graphs. Chart types would include Line, Scatter, Bar, Gantt, Area and Pie.

- HTML Edit Control (Both). Another control template, this one is a wrapper around the standard Microsoft DHTML control. It is a full WYWIWYG editor and supports creation of new files and editing/saving existing files. You can also optionally save/load from a data column.

- Rules Manager. This would let you specify business rules in the dictionary at the control/column level.

- Browse Width. Dynamic expanding/contracting listbox size.

- MCI Wrapper. A new wrapper around the Windows multi-media control, it would let you play/display multi-media files (AVI, WAV, etc.)

A Bit of SOAP

At the end of his last presentation on the last day of the workshop, Bob Zaunere pulled a little rabbit out of his hat. Almost as an afterthought, he did a brief demonstration of a Clarion SOAP client he had written. While neat, that's been done before (see Brian Staff's article here in Clarion Magazine). What I think most people missed was that he had also created the SOAP server in Clarion. If that sounds like gibberish to you right now, just trust me - that is a *really* great thing. SOAP and XML (along with an alphabet soup of other acronyms like UDDI, etc.) are the direction web services are going, and to be able to create a provider in Clarion is going to be killer.

Training (Online and CBT)

SoftVelocity is aiming to move in a big way to providing online web-based training. Details, pricing, exact topics (several were mentioned, including SQL), all are still to be determined, but Bob Zaunere is a big believer in it and wants it to happen.

The existing Foundations and Essentials Computer Based Training (CBT) courses are also slated to get an update for the newer Clarion versions (both are pretty much based on Clarion 5).

More Workshops

Due to the overwhelmingly positive response to this workshop, SoftVelocity plans to have more of them in the future (attendance greatly exceeded their expectations, as did the attendance to the Essentials and Mastery classes given before and after the workshop.) They mentioned possibly the Midwest and West coast of the U.S. (or

even Puerto Vallarta!), and also Europe.

Summing It All Up

In a very real way, Clarion 5.6 will be SoftVelocity's first Clarion release. Yes, they 'released' 5.5, but the feature set and code were inherited from TopSpeed -- this is the first one where SV has been in full control of the product's direction.

As such, I would like to point out a common thread that winds through almost every topic I've covered - one that I think is crucial to SoftVelocity's vision of the tools it builds (and that we all use): Virtually every item discussed is either leveraging current available technologies (ASP, ADO, COM), removing restrictions that have hindered Clarion (lack of true OS threading, no variant or Bstring datatypes) or is aimed at allowing easy integration with the newest upcoming standards (XML and SOAP, for instance). Unlike the latter day TopSpeed that had, in my opinion, developed a real we-know-what's-best-for-you attitude, SV seems focused with a laser-like clarity on the following goals:

- Listening to their customers. (YES!)
- Fully integrating Clarion with the rest of the Windows world rather than insulating us from it.
- Understanding and leveraging the core strengths of Clarion, such as the data dictionary and code generation. I haven't even mentioned their long-term goals of integrating/generating other languages (keep your eyes peeled - ASP and PHP are only the first examples.)

Bottom line: I left the conference totally stoked about the future of Clarion, the direction that SV is going, and all the new tools that they are making available. (And I got a nifty t-shirt as well)

Great job, SV, great job!

A longtime Clarion user, Tom Hebenstreit is an admitted tool junkie who refuses to go straight and code without his arsenal of third party products. During those rare moments when he isn't either using or writing about Clarion, he indulges his twin passions for blues and beer by performing around Southern California in a variety of totally-obscure-but-famous-any-day-now rock and blues bands.

Reader Comments

Add a comment

Thanks, I REALLY enjoyed your article! Not only the...
Thanks! The hardest part by far was trying to keep the...
Excellent summary. Thanks. The only question I still...