Main Menu | Log In | Subscribe | Links | FAQ | Advertising

Clarion MAGAZINE

Clarion
Development
Resources

published by
CoveComm Inc.

Main Page

Log In
Subscribe
Renewals

Frequently Asked
Questions

Site Index
Article Index
Author Index
Links To
Other Sites

Downloads
Open Source
Project
Issues in
PDF Format
Free Software

Advertising

Contact Us

Search

Issue Index

February, 2000

Clarion 5.5 Beta 2 To Include iBuild Features

TopSpeed has released the feature list for 5.5 Beta 2 and it's a long one! Topping the list: almost all of the work done for iBuild has been incorporated into 5.5.
(Feb 8,2000)

Edit-In-Place:Lights, Camera, Action...Take Two!

There's more than one way to add edit-in-place to a browse. James Cooke lays the groundwork for making a form look like EIP. Part 1 of 2.
(Feb 8,2000)

Review: MessageEx and SysAni

Tom Hebenstreit reviews two new products from a new Clarion third party vendor. SysAni adds standard and custom animations to your apps, and MessageEx enhances the standard Message() function.
(Feb 8,2000)

Skeleton Basics IV: List Variations

Web Builder HTML list boxes are quite different from the Java-enabled list boxes of CWIC1. Steve Parker explains how to change the appearance of these list boxes.
(Feb 8,2000)

Attend ETC, Win Stuff!

The East Tennessee Clarion Conference isn't just about attending sessions - it's also about door prizes! The list of sponsors keeps growing; products for raffle include: ClarionMag 2 year subscription; Automated Fax Engine; Report/Presentation Manager; ABC SQL Template Set; Translator Plus, Enterprise Edition; Creative Reporting Tools for C5; File Manager 2; Special Agent; NetTalk; EzHelp; Makeover; TearOff; XLIB Library with full source; Imaging Templates; UltraTree Platinum; LogFlash; SearchFlash; Linder SetupBuilder 3.0; LSPzip Data Compression Toolkit; Clarion Source Search; CW Assistant; Super Template of Choice; Clarion Companion; and probably lots more!
(Feb 8,2000)

Skeleton Basics V: Listbox Relocation

In this fifth installment in the Skeleton Basics series Steve Parker shows how to relocate list boxes on the web page.
(Feb 15,2000)

Edit-In-Place:Lights, Camera, Action...Take Two, Part Two!

There's more than one way to add edit-in-place to a browse. James Cooke lays the groundwork for making a form look like EIP. Part 2 of 2.
(Feb 15,2000)

The Clarion Advisor: Redirection Files Revisited

John Power shows how to use a custom redirection file to optimize data backup.
(Feb 22,2000)

ToolTalk: Needles And Haystacks
Tom Hebenstreit goes searching through the World Wide Haystack and threads together a list of useful Clarion sites.
(Feb 22,2000)

Are Accounting Objects Possible?
Is it possible to write a set of objects to manage most accounting software needs? David Podger sets some requirements and launches an open source project.
(Feb 22,2000)

EIP Template Updated
James Cooke has provided a bug fix update to his EIP template. The other files in the zip are unchanged.
(Feb 22,2000)

Last Chance To Renew And Save!
Our first anniversary special is about to end - renew now and save!
(Feb 29,2000)

Last Chance To Save On A New Subscription!
Our first anniversary subscription sale is about to end! The subscription rate is going up to $80/year on March 1, but you can still subscribe now for $75 and we'll give you one month free. That's a 13% savings. Or subscribe for two years and get 30% off your second year. With either option you can add all the back issues (March '99 to January '00) for just $50 (save $16). Buy a two year subscription with all the back issues and save $46!
(Feb 29,2000)

February 2000 News
Clarion news, notes, and happenings from around the globe.
(Feb 29,2000)

The Novice's Corner: Clarion Code 4
In part 4 of this series Dave Harms looks at the how and why of list box formatting.
(Feb 29,2000)

List Box Marking
Professor Parker gets out his marking pen and explains how list box marking works, and how it can be made better.
(Feb 29,2000)

Open Source Update: Thread Manager
This is a very cool thread manager which is extremely easy to use. It's also unusual in that it allows you to completely bypass the usual START() parameter limits. With this thread manager you can pass *any* data type to the START()ed procedure.
(Feb 29,2000)

Main Menu | Log In | Subscribe | Links | FAQ | Advertising

Clarion MAGAZINE

Clarion Development Resources

published by CoveComm Inc.

Main Page

Log In
Subscribe
Renewals

Frequently Asked Questions

Site Index
Article Index
Author Index
Links To Other Sites

Downloads
Open Source Project
Issues in PDF Format
Free Software

Advertising

Contact Us

Search

Feature Article

February, 2000

Clarion 5.5 Beta 2 Features

The following message was posted in numerous TopSpeed newsgroups on Tuesday, February 8th. It is reproduced here unaltered, in its entirety.

From: "Sean Caputo" <scaputo@topspeed.com>
Newsgroups: topspeed.products.c5ee
Subject: beta 2 White Pages
Date: Tue, 8 Feb 2000 17:57:35 -0500

At last we can talk about the long-awaited beta2 release of Clarion5.5. What have we been doing for the last 4 months? Well at DevCon we announced a new product in the works called iBuild. The iBuild product, intended as a separate product from Clarion was well received by all of the attendees, with one universal request -- make it compatible with Clarion 5.5. In the months since DevCon and the release of 5.5 beta1 we continued to hear from customers anxious to see an early release of iBuild. Everyone who we spoke to asked us to consider making iBuild work within Clarion 5.5.

We listened, we thought it over, and we agreed. Almost all of the development work done for iBuild is now incorporated into the Clarion 5.5 beta2. The components of iBuild which are specifically internet related have been packaged into the "iBuild Toolkit". As a reward to our beta program participants, we are bundling the beta version of the iBuild Toolkit with this release. Customers who purchase 5.5 while still in beta will receive the gold release of the toolkit free of charge. You will also receive any additional components we add to the iBuild toolkit. After 5.5 is released the toolkit will be packaged and sold separately.

One final note, this is only an overview of the major enhancements to 5.5. As we complete the Help and Documentation we will update the 5.5 Web Resource Center with information on additional features. We have completed documentation on the new language statements, driver functionality, and templates and their supporting class libraries. We are still working on the web class library documentation, and will post as soon as possible. Online Help will also be made available.

Was it worth the wait? We think so, and we think when you have had a chance to look at the features and enhancements we packed into beta2, that you'll agree too.

Regards,

Robert Zaunere VP Product Mgt. TopSpeed Corporation

Clarion 5.5 Beta 2

This beta release continues the polishing of the development environment, adds powerful enhancements

Clarion 5.5 Beta 2 To Include iBuild Features
(Feb 8,2000)

Anniversary Subscription Special!
(Feb 8,2000)

Renew Your ClarionMag Subscription And Save!
(Feb 8,2000)

Edit-In-Place:Lights, Camera, Action...Take Two!
(Feb 8,2000)

Review: MessageEx and SysAni
(Feb 8,2000)

February 2000 News
(Feb 8,2000)

Skeleton Basics IV: List Variations
(Feb 8,2000)

Attend ETC, Win Stuff!
(Feb 8,2000)

to database access, and adds a number of new templates and their supporting class libraries. The combination of these elements facilitates the creation of feature-rich applications for either the desktop or the Internet. In addition, there were numerous fixes made to the runtime library and the development environment.

The most significant addition to this beta is the iBuild Toolkit. We are including these components as a reward to our Beta Program participants. As a Beta participant, you will receive a beta version now and the "gold" release later.

Enhancements to the development environment

The Property Editor

Beta 1 introduced the Property Editor in the Dictionary Editor, Window Formatter and Report Formatter. Based on your feedback, we have further enhanced the Property Editor.

You can now multi-select controls in the visual formatters before calling the Property Editor and edit properties for the controls you selected. In addition, any controls selected in the Property Editor now remain selected in the Window or Report formatters when you close the Property Editor. The Property Editor now has an Apply button, which allows you to implement changes to control properties and remain in the Property Editor. The changes immediately display on the underlying Window or Report allowing you to see the result of your changes. The Property Editor in the App Tree now retains its position and the level of expansion. The Property Editor in the Dictionary Editor now provides access to all appropriate attributes.

File Driver Enhancements

Callbacks

New Clarion language statements have been added to allow registering and unregistering Callback Interfaces. This provides the equivalent of client-side SQL triggers for either ISAM or SQL files.

Improved syntax support for SQL filters

You can now use SQL code within FILTER, ORDER, or JOIN, expressions Anything specified in the sql expression is treated as pure SQL code.

Examples:

MyView VIEW(AFile), FILTER('Ord:PurchaseDate = TODAY() AND ' & | 'SQL(A.Info LIKE ''%Must deliver today%'')'), ORDER('SQL(A.Customer)') END

MyView{PROP:Filter} = 'SQL(A.Customer IN (SELECT CustID FROM Customers WHERE BadCustomer = 0)'

Stored Procedures in SQL files

Support for calling stored procedures has been enhanced to add support for output parameters and return codes. Now, in addition to getting a result set, you can receive return codes (such as an error code) or an output parameter for a stored procedure.

Improved ABC support for NULLs in SQL tables

Additional File Properties

A number of file properties (such as PROP:Record) are now available to allow you to get a record group directly from a file. This allow you to write generic file code.

Example:

Rec &GROUP Customer FILE,DRIVER('TopSpeed') Record RECORD Name STRING(20) ...

CODE Rec &= Customer{PROP:Record}

Template Enhancements

View-only Form

Database developers often find the need to display a record in read-only mode. While this was relatively simple to accomplish using embedded source code, we know that templates can make the task easier. For that purpose we have added three new templates to support view-only forms. When used over the Web, read-only controls display as strings.

The Configure View Only Mode Form extension template can be added to any form and allows you to specify what to do with a control when the form is opened in view-only mode. You can set a control's view action to none, disable, read-only, or hide. The Browse View button control template can be added to any browse to call its form in view-only mode. It simply sets the Record Action to ViewRecord. The View-only form Procedure template is similar to the standard form template but has the Configure View Only Mode Form extension template pre-populated for your convenience.

Record Auditing

Keeping track of changes to a data set is an important aspect of database applications. The DbAuditing global extension template allows you to specify Files and particular Fields in a file to audit and generates a customizable ASCII Log file to keep track of the changes.

Command Line Options

A new global extension-RunCommandLineProc--allows you to specify which procedure to run at startup based on a command line parameter. This allows you to have more than one "main" procedure and actually compile more than one program into a single program.

Fuzzy matching

This control template populates an ENTRY control and BUTTON control to support weighted searches and sorting of the result set in order of relevance. Merely add this extension to a BrowseBox control and end users can easily search the data and see the results in the order you specify.

Browse Grid

This extension to a BrowseBox allows you to display controls in a matrix format, dependent upon the size of the BrowseBox LIST control. By adjusting the size of the BrowsGrid control and the size of its parent LIST control you can display data in a single column or multiple columns. Adding the Dynamic Image control allows you to have clickable thumbnails that can perform any action associated with a control on your window.

This is particularly useful for Web applications, but can extend the capabilities of a desktop application, too.

Redirect to page

A simple code template that allows you to dynamically change the page to return (URL) upon exit of a Web application.

The iBuild ToolKit

App Templates

AppGuard

This template provides an easy way to limit access to applications at the procedure level or at the application level. Another control template allows new users to login and be added to the data file. The AppGuard templates are highly configurable and easily tie into an existing database.

AppHitManager

This template provides an easy way to record the number of accesses (hits) to applications or procedures in them. Hit counts can be tied to a particular window activity or control. Another control template allows you to display the count.

Cybercash Templates

This set of templates allows you to use CyberCash for credit card verifications and authorizations.

CyberCash is a world leader in e-commerce technologies and services, enabling commerce across the entire market spectrum from electronic retailing environments to the Internet. CyberCash provides a complete line of software products and services allowing merchants, billers, financial institutions and consumers to conduct secure transactions and other e-commerce functions using the broadest array of popular payment forms. Credit, debit, purchase cards, cash, checks, smart cards and alternative payment types (e.g., "frequent buyer" or loyalty programs) are all supported by CyberCash payment solutions. More information on CyberCash can be found at www.cybercash.com.

These templates allow you to easily add functionality to:

Capture and authorize a payment Configure your CyberCash account

Messaging Templates

This set of templates easily adds email (SMTP) transfers or newsgroup (NNTP) postings from your database. These templates incorporate the code needed to turn your application into a complete Mail or News client and works independently of any other Mail or News client.

These templates allow you to easily add functionality to:

Send a document (email or newsgroup posting) Merge data into documents before sending Send messages upon certain events (triggers) Establish global recipients and groups of recipients Add an attachment file to a message Transmit a message to a single recipient or a group of recipients using a process. Send the same message to newsgroups and email in a single pass. The underlying technology for the messaging library provides the basis for additional features which may be added later. These include FTP and Telnet capabilities.

Shopping Cart Templates

These templates are designed to work in a specific type of application : a Web-based Shopping Cart. Used in conjunction with the supplied data dictionary, you can use these templates to create your own custom shopping cart.

These templates support most of the features popular in current shopping cart applications such as:

Adding a record to a cart Viewing an active cart Allowing discounts based on distinct criteria Processing an order Computing Sales tax Computing Shipping charges Generating an invoice

Main Menu | Log In | Subscribe | Links | FAQ | Advertising

Clarion MAGAZINE

Clarion Development Resources

published by CoveComm Inc.

Main Page

Log In
Subscribe
Renewals

Frequently Asked Questions

Site Index
Article Index
Author Index
Links To Other Sites

Downloads
Open Source Project
Issues in PDF Format
Free Software

Advertising

Contact Us

Search

Feature Article

February, 2000

Clarion 5.5 Beta 2 To Include iBuild Features
(Feb 8,2000)

Edit-In-Place:Lights, Camera, Action...Take Two!
(Feb 8,2000)

Review: MessageEx and SysAni
(Feb 8,2000)

Skeleton Basics IV: List Variations
(Feb 8,2000)

Attend ETC, Win Stuff!
(Feb 8,2000)

Edit-In-Place: Lights, Camera, Action... Take Two!

by James Cooke

Edit-in-place, or EIP, has come a long way since CW20 days, where it was the standard practice to hand code the entire process using prop:edit. I have been a template adherent since early DOS days, and have therefore developed a healthy loathing for unnecessary hand code. To me, the hallmark of a "nice" application is one with very few or no embeds, and with all the extra functionality wrapped up in nice, predictable template interfaces. So when ABC EIP classes were introduced, Clarion got that much sweeter for me - to an extent . All things being equal, EIP is not quite as RAD as some would like it to be.

Limitations Of EIP

The following points are limitations I have found using the current paradigm of EIP, bearing in mind though that the current ABC implementation of EIP still outstrips the functionality and grace of most tools' EIP components:

- Separate encapsulation of data browsing and data modification is gone; all logic exists in one procedure with the result being often horribly complex.
- Validations which are usually a piece of cake in standard update forms due to convenient ABC and third party control, extension and code templates, are no longer available using EIP, so programmers have to revert to the stone age and do most of the validations by hand.
- Embed points that are available for each entry, droplist or check control are no longer there to the degree some would like them to be.
- Validations based on the contents of another field have also become more complex; the convenience of using prop:AcceptAll to validate all other fields before saving the record is gone.
- Editing of fields in the table is limited to the columns displayed in the listbox only.

On the other hand, EIP is a very intuitive way to update records from a browse. Users

DeveloperPLUS
Your source for development tools and add-ons.
Your outlet for application sales.

like EIP, because they associate the modification of a record in a browse with the browse window itself, and not some other window that pops up!

Limitations Of The Browse-Form Approach

The traditional form of doing updates in the Clarion environment is Menu-Browse-Form. The main complaint with this method is that for every update a new procedure is added to the program, meaning more clutter and another non-standard, illogical learning curve for the user. Easy for us as programmers, but in the end the user rules!

It is worth noting that some of the problems of EIP are acceptably addressed by Clarion's Menu-Browse-Form method. Database maintenance is kept simple, with all the logically separate parts cleanly isolated from one another in the code. It is this very crisp definition of separate database functions that allowed many a newbie (including myself) to grasp the fundamentals of relational database theory. It was the advent of EIP incorporated into the browse module that has caused many a mile of spaghetti code. On the other hand, while menu-browse-form is convenient and simple to the programmer, it is not that "user friendly" (apologies for using that dreadful cliché!).

The obvious conclusion would be to aim for the intuitive look and feel of EIP *as well as* the use of a separate, standard update form. The purpose of this article is to explain a very simple but workable way to do this.

The Framework

The fundamental idea is to maintain the standard browse in its original form, and dynamically "reshape" the update form to "fit over" the browse box at the correct position, height and width. This would have to take into account a plethora of attributes about the state of the listbox - the number of columns, the font, the currently selected row, the width of each row, resizing of columns, whether there are headers or not, and more. The problem is that these local values are usually out of the scope when the update form has opened. The number of variables can also vary, as there can be a variable number of columns in the listbox. Here are some options that can address these problems.

- Save the properties of the listbox as threaded globals and the column variables in a global queue. This will work, except the overhead of all these globals could get to be too much in a large application, as each browse would require its own set. In addition, it would be very untidy and difficult to maintain.
- Put the browse and form in the same module and save the properties of the listbox as module data and the column variables in a queue. This would be better than the first option, because of the encapsulation of the properties within the module. Putting the browse and form in the same module might not be desirable to some.
- Use a global reference variable to refer back to the controls of the browse window. This is better than the first two options, but global variables should in principle be avoided where possible.

All of these methods will work but are clunky, and a template written to automate the process might be unnecessarily complex. It is clear that a more elegant solution would be required for a real life application.

Property Assignments: A Question Of Scope?

A few years ago I determined that it might be a useful exercise to spend some time in another camp. Learning Visual Basic was an interesting, albeit frustrating exercise. Without climbing into an longspun diatribe about the merits and demerits of the language, there was one very flexible aspect of programming using Visual Basic: all window properties with their methods, control properties with their methods and even procedures are accessible from other windows and procedures. It appears that all of these variables have global scope. To change a control's properties or methods using VB, all one does is qualify the assignment statement with the window and control label:

```
winCustomer.lstCuslist.Caption = "Customer List"
```

and

```
winCustomer.lstCuslist.MoveLast()
```

This means that in VB you are able to determine and even modify the state of controls from a completely separate procedure in a separate module. That would be the first step in this EIP idea - making the properties of the listbox accessible to the update form.

Clarion does not allow this as directly... or does it? That was the question raised at a recent Cape Town user group meeting, and the answers that came out of the discussion were certainly enlightening. It was pointed out that the scope for property queries and assignments is not limited in terms of procedural or modular encapsulation, but rather within the structure of the last active window. Therefore, *unless another window has been opened in the interim,* the properties of the last accessed window are in scope.

What has that to do with EIP? It means that *before an update form's window has opened,* all the properties of the browse window and its controls are accessible to the update form. This is useful because the traditional reference variable can be discarded. Try the following on a simple app:

Embed this in a browse window's button:

```
0{prop:text}='Changed locally'
```

The result will be that the browse's caption will read, "Changed locally"

Embed this in the update form's `Init` embed point, before opening the window:

```
0{prop:text}='Changed from somewhere else'
```

The result will be that the browse's caption will read "Changed from somewhere else."

This works only because the scope of property assignments refer to the last active window, and *the actual control number and not the equate is used in the assignment statement.*

Referencing Out-Of-Scope Controls

In Clarion, a reference to the properties of a local control is done by prepending the label of the control with a question mark. The equate label is resolved internally to the control number, and the control number represents the physical order in which the control is declared in the screen definition source code. Since the equate label is a local variable, a compiler error will result if you try to address the control by its equate label from a procedure other than the one containing that control.

Therefore it is necessary to refer to the control by using the actual control number that the equate represented. The problem there is that this number is internally assigned and can change depending on the order and number of controls in the window. One way to resolve this is to force the control to have a certain control number, and address the control number by this number and not its equate label. To force the control number, append the original equate with a comma followed by the desired control number, for example:

```
?button1
```

will become

```
?button1,1000
```

Hence in the above example, "1000" should be used as the equate instead of ?button1

```
TheColor = window$1000{prop:color}
```

is used instead of

```
TheColor = window$button1{prop:color}
```

Warning: overriding field equate labels may clash with other third party templates that are doing the same thing. If there is a conflict, just make the number much bigger, i.e. 10000.

That's the theory. In the next week's issue I'll show you how to go about using these techniques to make your update form work like EIP.

Read Part 2

James Cooke is currently using Clarion and Informix to develop client server solutions for the fruit export industry. He started in programming in 1991 by writing a CPD POS app for his "hardware store" on a fleamarket. He sold it to a hardware shop, and from there was sucked into the IT industry by developing a plethora of apps for several small companies. He spends his time reading, listening to classical music, scuba diving and hiking. He lives in Cape Town, South Africa with his wife and "zoo."

Clarion MAGAZINE

Clarion Development Resources

published by CoveComm Inc.

Main Page

Log In
Subscribe
Renewals

Frequently Asked Questions

Site Index
Article Index
Author Index
Links To Other Sites

Downloads
Open Source Project
Issues in PDF Format
Free Software

Advertising

Contact Us

Search

Feature Article

February, 2000

Clarion 5.5 Beta 2 To Include iBuild Features
(Feb 8,2000)

Edit-In-Place:Lights, Camera, Action...Take Two!
(Feb 8,2000)

Review: MessageEx and SysAni
(Feb 8,2000)

Skeleton Basics IV: List Variations
(Feb 8,2000)

Attend ETC, Win Stuff!
(Feb 8,2000)

Solid Software MessageEx 1.2 and SysAni 1.1

Tom Hebenstreit, Reviews Editor

In this review I will be covering two simple but useful products from a new vendor on the Clarion scene – Solid Software. Before I start, though, I'd like to make it very clear that even though I am covering both products in one review, they are completely separate, standalone products. Neither one needs the other to work.

The first product is MessageEx, an expanded replacement for the standard Clarion Message() function. Think of it as Message() on steroids. Lots of steroids.

The second is SysAni, and it lets you add both custom and standard Windows animations to your programs. What does that mean? Well, you know all those cute little papers flying from here to there when you copy, move or delete files in Windows Explorer? Those are known as "common" animations, and are actually AVI (Audio Video Interleaved) files being displayed on the progress window by a built-in Windows control. SysAni lets you do the same in your Clarion programs without having to delve into the arcane depths of the Windows API.

Major Features: MessageEx

MessageEx is single new function combined with a simple-to-use code template that lets you set the parameters for the function call. Features beyond the standard Message() function include:

- Display custom images (even animated gifs), linked-in as well as on-disk files
- Specify a custom font (and charset)
- Create your own buttons (not just standard Yes, No, OK, Cancel, etc.)
- Display a check boxes with user-defined text
- Show wallpaper images
- Choose background colors

- Use flat buttons
- Play sounds
- Display "timed messages" (with or without buttons)

In other words, you can do a *whole* lot more than just the usual pop-up message. It should also be noted that MessageEx does not interfere with the standard Message() function in any way. There is nothing to stop you from using both of them in the same application.

Major Features: SysAni

Behind the template, SysAni is a Clarion wrapper class that makes it easier to use the built-in Windows animation control. Capabilities include:

- Display both standard Windows animations and custom animations
- Options for setting transparency, starting point, auto start and more
- Utilizes native Windows threads

Both products are supplied in LIB and DLL form, and are compatible with Clarion 5 ABC and Legacy. For the adventurous, the Clarion 5.5 beta is also supported. Source code is not included.

Installation

MessageEx and SysAni can be downloaded from the Solid Software web site, as can a standalone demo for each product. File sizes range from 275K to just under 400K, depending on the item.

I downloaded all four installation files, and began by installing the demos to get a feel for what each tool could do (more on the demos themselves later). Each install was clean and straightforward, automatically finding my Clarion 5 folder and installing the various bits and pieces in the appropriate sub-folders. For the actual product installations, they also displayed the documentation at the end (always a plus in my book).

One touch that I particularly liked was that the installations used at least some of the guidelines for third-party products from the Clarion 3rd Party Association (C3PA). In this case, each of the four installs defaulted to placing itself under my existing Clarion 5 Accessories program group. When you install as many third party products as I do, you quickly grow to appreciate anything that helps control bloat on the Start Program menu.

One nit that I noticed was that each of the installs displayed invalid information for the amount of disk space required and the free space left on the drive. According to the instructions on that dialog box, my drives did not have enough space to install. Seeing as how the drive actually had approximately ten gigabytes of free space on it, I just ignored the instructions and everything installed perfectly.

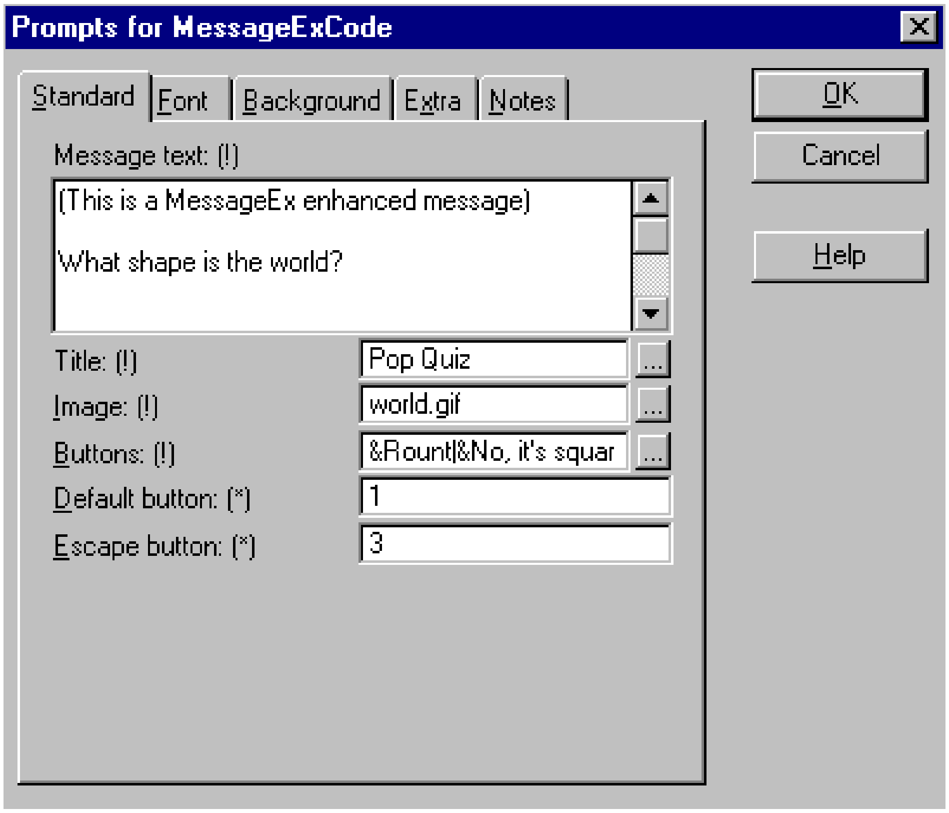
Implementation

As usual, the first step was to register the templates (both ABC and Legacy template chains are provided for both products). Next, I needed to add a global extension template (one each). There is only one option on either extension, and that is to specify the link mode to be used (LIB or DLL). If you choose DLL, you'll need to remember to ship the appropriate DLL with your program.

MessageEx

For MessageEx, all that is left is to simply use the supplied code template in the appropriate embed point wherever you want to display extended messages. If you prefer to call the MessageEx function directly in your source or embed code, there are up to 21 parameters that can be used to specify MessageEx's behavior (yikes!). Fortunately, all are well documented. Figure 1 shows some of the template options for controlling MessageEx.

Figure 1. MessageEx code template Standard options



Message text: (!)

(This is a MessageEx enhanced message)

What shape is the world?

Title: (!) Pop Quiz

Image: (!) world.gif

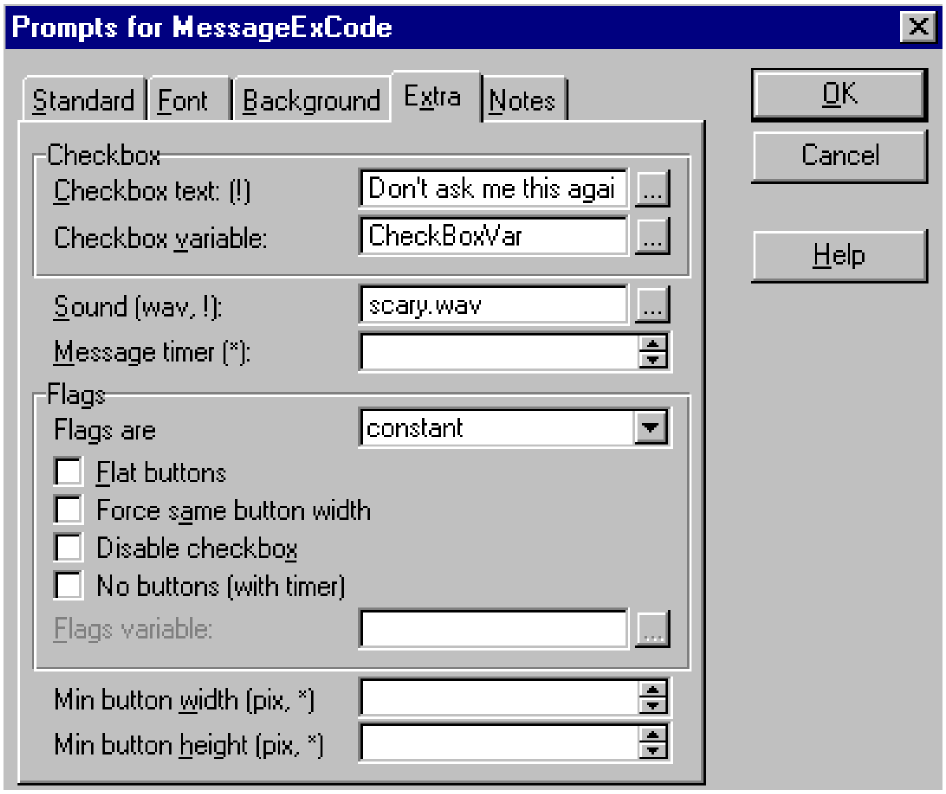
Buttons: (!) &Round|&No, it's squar

Default button: (*) 1

Escape button: (*) 3

Figure 2 shows a few of the extra options for check boxes, sounds and more.

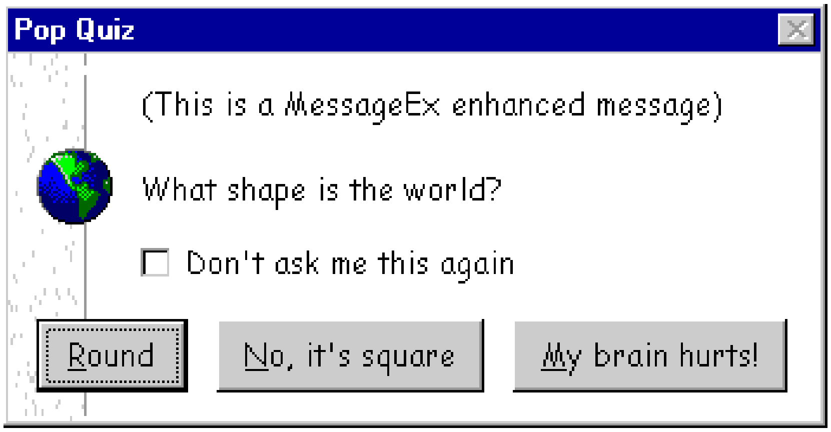
Figure 2. MessageEx code template Extra options



Checkbox
Checkbox text: (!) Don't ask me this agai
Checkbox variable: CheckBoxVar
Sound (wav, !): scary.wav
Message timer (*):
Flags
Flags are constant
□ Flat buttons
□ Force same button width
□ Disable checkbox
□ No buttons (with timer)
Flags variable:
Min button width (pix, *)
Min button height (pix, *)

After setting more options for the font and background image on the template, I ran my program and got the result shown in Figure 3.

Figure 3. The resulting MessageEx popup



(This is a MessageEx enhanced message)

What shape is the world?

□ Don't ask me this again

What you can’t tell from the above image capture is that the globe is actually a spinning animated GIF file, and that a sound file is played when the message pops up. (Yes, Virginia, this means that you too can now create popup messages that are not only annoying, but garish and noisy as well.)

In case you are curious about what’s happening behind the scenes, here is the generated code which produced the above message:

Figure 4. MessageEx generated code for the message in Figure 3.

```
MsgExButton# = MessageEx( |
 '(This is a MessageEx enhanced message)<13,10,13,10>’ &|
 ‘What shape is the world?',      |  !Text
 'Pop Quiz',                      |  !Title
 'world.gif',                     |  !Image
 '&Round|&No, it''s square|My brain hurts!',   |  !Buttons
 1,                               |  !Default button
 3,                               |  !Escape button
 'Don''t ask me this again',      |  !Checkbox text
 CheckBoxVar,                     |  !Checkbox variable
 'Comic Sans',                    |  !Fontname
 10,                              |  !Fontsize
 0,                               |  !Fontcolor
 ,                                |  !Fontstyle
 CHARSET:ANSI,                    |  !Charset
 -1,                              |  !Backcolor
 'C:\Temp\wallpaper.gif',         |  !Wallpaper image
 0,                               |  !Wallpaper mode
'scary.wav',                      |  !Sound
,                                 |  !Options (flags)
,                                 |  !Min button width
,                                 |  !Min button height
)                                    !Timeout
```

Note the use of the `MsgExButton#` implicit variable to receive the button choice made by the user. You use that variable to determine your own course of action in your embed code following the MessageEx code template.

My one big request for MessageEx is to have a utility (similar to the demo) that would let me interactively set and test all of the various options. Then when everything was right it would paste the generated code to the clipboard for easy placement in source.

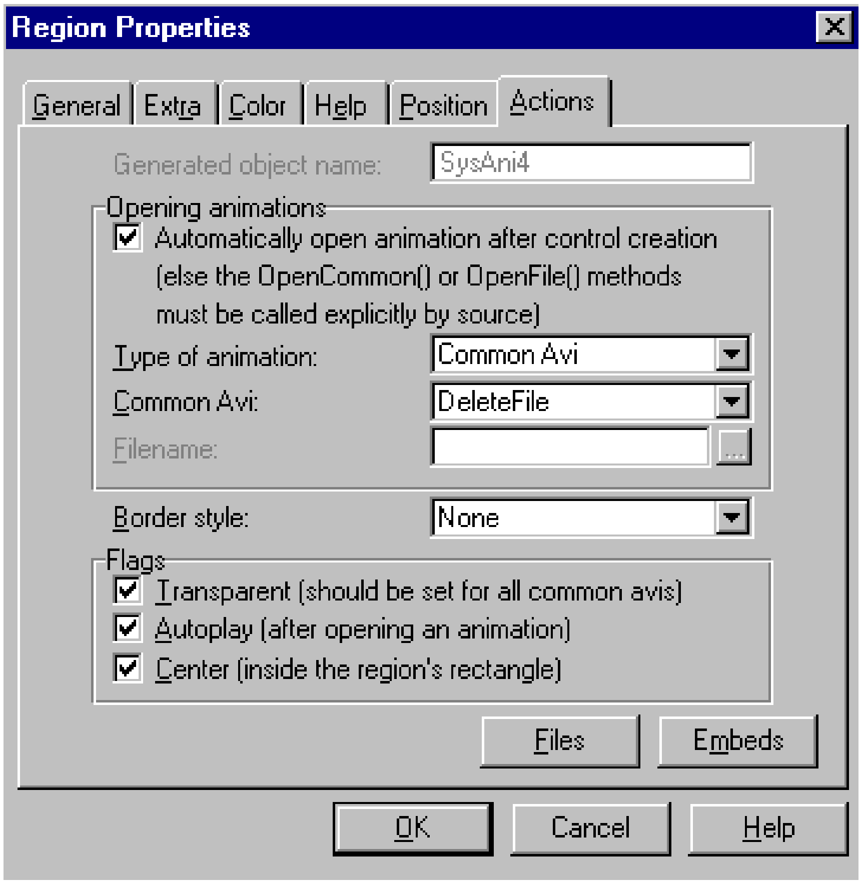
Why do I want this? Well, while the template works fine in embeds, it doesn’t work as well for source procedures or within a larger block of pure source code. In those cases, you are stuck with having to create code similar to Figure 4 by hand (all 21 parameters of it), or break up your embed code into separate points, or place the code template in a routine.

All in all, though, I found it extremely easy to add MessageEx to an application.

SysAni

SysAni is also easy to use. As you would expect, in this case the template is a control template and so it must be placed on a window. Once it is there, you use the options on the Actions tab to tell SysAni what to do. Example:

Figure 5. The SysAni control template Actions tab



Generated object name: SysAni4

Opening animations

☑ Automatically open animation after control creation (else the OpenCommon() or OpenFile() methods must be called explicitly by source)

Type of animation: Common Avi

Common Avi: DeleteFile

Filename:

Border style: None

Flags

☑ Transparent (should be set for all common avis)

☑ Autoplay (after opening an animation)

☑ Center (inside the region's rectangle)

As you can see, there are far fewer options to set here than for MessageEx.

In the above example, SysAni will display the common (built-in) delete animation where the little pieces of paper travel from left to right, starting in a folder and then disappearing as they move across the animation. Other common animations include emptying the Recycle Bin, the circling magnifying glass/computer displayed on the standard Find Files or Folders dialog, and the flashlight/folder combination displayed when Windows is searching for a Help file.

Performance

Both products performed admirably. I found MessageEx to be very useful for making messages both stand out (not the usual bland gray box) *and* blend in (they could be matched with the overall visual scheme of the application). Judicious use of color in the fonts and background can also really help to distinguish one message type from another.

The icing on the cake is the ability to add a checkbox to your dialogs. I can’t tell you how many times I have wanted to add a "Don’t show this again" type of option to various messages and have either had to go without or create extra Window procedures by hand.

The only extra feature that I would like to see in MessageEx would be the option to close timed button-less dialogs early by, for example, pressing the Escape key.

As for SysAni, performance is excellent, primarily due to the nature of the underlying animation control. Being native to Windows itself, it uses a true Windows thread as opposed to a Clarion style thread. Because of this, the animations are always smooth, especially in situations where an animated GIF can suffer from flicker or even stop working due to the Clarion program being busy or modal.

It should be noted that even though AVI files can contain audio, the Windows animation control does not support audio (it simply ignores it). This is not a fault with SysAni, but a restriction of the control itself.

Documentation

The documentation for each product is provided in the form of an HTML file (i.e., a web page you view in a browser). I've become a fan of this style of documentation lately (at least for smaller products), as it provides for both easy printing and easy movement within the documentation via embedded links.

Both documents are well written, and do a good job of covering both the use of the templates and of documenting how to use the products via source code. There are no Windows help files for context sensitive help within the templates, but both products are so simple to use that I didn't find that a drawback.

Special mention should be made of the MessageEx and SysAni demo programs. Both are a model of simplicity and clarity, not only showing the capabilities of the products, but letting you interactively test them out by entering and changing the full range of options.

The only real improvement I would suggest would be for the SysAni documentation to start off with a bit of background on what the Windows animation control actually is and how it works. That would help put the following instructions into perspective.

Technical Support

Technical support is offered via email, and it appears to be very responsive. Even when I asked a question on a Sunday, I still received an answer the same day.

I must admit, though, that I had a hard time thinking up questions to ask - there just isn't a whole lot of complexity involved in using either product.

Summary

Both MessageEx and SysAni are simple to use, perform exactly as advertised, and have a polished, quality feel to them. If you have even the slightest interest in either product, I highly recommend you check out their excellent demonstration programs.

For myself, I would not hesitate to use either one in my own programs. As you can probably guess from my delight with the checkbox option, MessageEx has already earned itself a place in my own standard set of tools.

It is also a pleasure to see quality new vendors join the Clarion third party market, and I look forward to any other products Solid Software chooses to release.

PRODUCT RATING	
Overall	▲▲▲◭
Ability to do the task	Excellent
Ease of use	Very Good
Ease of installation	Very Good
Documentation	Very Good
Technical support	Very Good
Black box DLLs/LIBs	Yes

LEGEND	
First class all the way	▲▲▲▲
More than adequate	▲▲▲
Barely adequate	▲▲
Don't even think about it	▲

MessageEx 1.2 is available directly from the Solid Software web site for 49 Euro. SysAni is listed for 29 Euro. As you might infer from the pricing, Solid Software is based in Europe – Denmark, in fact. For the exchange-rate-challenged among us (myself included), one Euro is roughly equal to one U.S. dollar as of this writing.

Orders can be placed via the web using Visa or Mastercard, but as of the date of this review they did not yet have secure ordering set up (an oversight that I hope will be fixed soon).

For more information, or to download the demos, visit the Solid Software web site at: http://www.solidsoftware.de

Reviewed by Tom Hebenstreit

Main Menu | Log In | Subscribe | Links | FAQ | Advertising

Clarion MAGAZINE

Clarion
Development
Resources

published by
CoveComm Inc.

Main Page

Log In
Subscribe
Renewals

Frequently Asked
Questions

Site Index
Article Index
Author Index
Links To
Other Sites

Downloads
Open Source
Project
Issues in
PDF Format
Free Software

Advertising

Contact Us

Search

Feature Article

February, 2000

Skeleton Basics IV: List Variations

by Steve Parker

Internet Connect provided Java-based list boxes by default. These lists looked and behaved very much like standard Windows list boxes, and the apps produced by Internet Connect were fairly described as "Windows on the Web." Anyone familiar with working a Windows app could use one without special instruction. The trade-offs were that these Java-based list boxes required downloading a good-sized set of Java classes, and they scrolled slowly.

WebBuilder does away with that client-side Java. Entirely.

List boxes are now rendered in standard HTML with just enough JavaScript (not Java) to allow for calling an update form and getting the expected record back. The trade-off this time is that list boxes don’t actually scroll: the records stay in place and an indicator moves. That indicator is a radio button instead of a full-width selector bar (color coding the selected record is also possible). There are no scroll bars, horizontal or vertical. Also, icons cannot be supported as they required Java (.ICO is not a format supported by browsers).

In fact, these are not lists in any sense that we normally understand that term. Instead, they are an HTML representation of the state of the browse control. The queue from which the list box is formatted has actually been created and updated on the server; the entries in it format the HTML representation.

Because there are no scroll bars on the HTML list box and there may not be a toolbar, there is no assurance that the user is provided with a way to navigate the list. Therefore, a set of navigation buttons are supplied by the skeletons; these buttons are placed below the list area.

If the app does have a toolbar and it has not been marked "Hide when launched from browser" the result looks something like Figure 1 (both sets of navigation buttons work,

Clarion 5.5 Beta 2 To Include iBuild Features
(Feb 8,2000)

Edit-In-Place:Lights, Camera, Action...Take Two!
(Feb 8,2000)

Review: MessageEx and SysAni
(Feb 8,2000)

Skeleton Basics IV: List Variations
(Feb 8,2000)

Attend ETC, Win Stuff!
(Feb 8,2000)

Skeleton Basics Part 1

Skeleton Basics Part 2

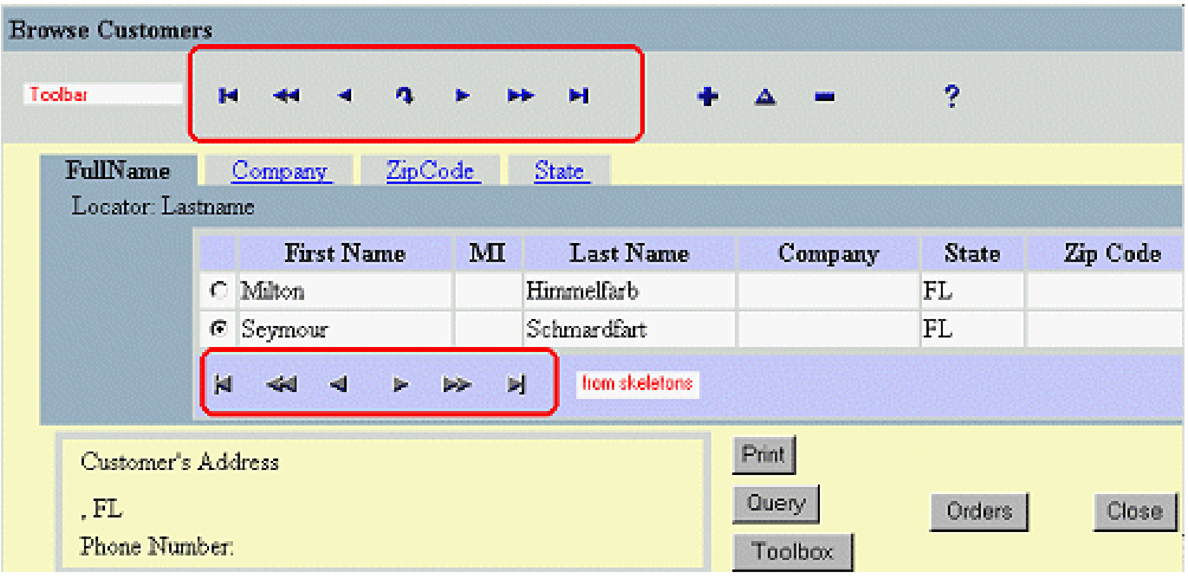
Skeleton Basics Part 3

Skeleton Basics Part 4

Skeleton Basics Part 5

by the way).

Figure 1. Double toolbars on the browse page.



	First Name	MI	Last Name	Company	State	Zip Code
○	Milton		Himmelfarb		FL	
●	Seymour		Schmardfart		FL	

Customer's Address

, FL

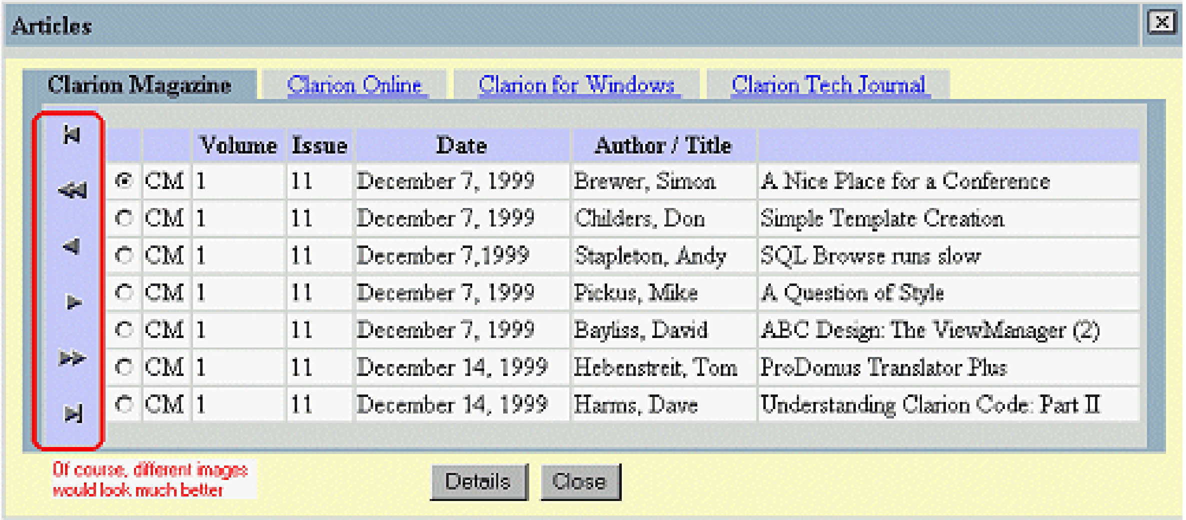
Phone Number:

Figure 1 is taken from the example app, Inventory.APP, which came with Clarion5.

Now, the obvious question is, if I do use a toolbar on my frame, do I really need or want a second set of navigation buttons? And whether I have toolbar navigation buttons or not, do I want the skeleton-supplied set of buttons below the list area?

Can’t I have the skeleton-supplied buttons on the side? Something like Figure 2?

Figure 2. A browse with navigation buttons on the side.

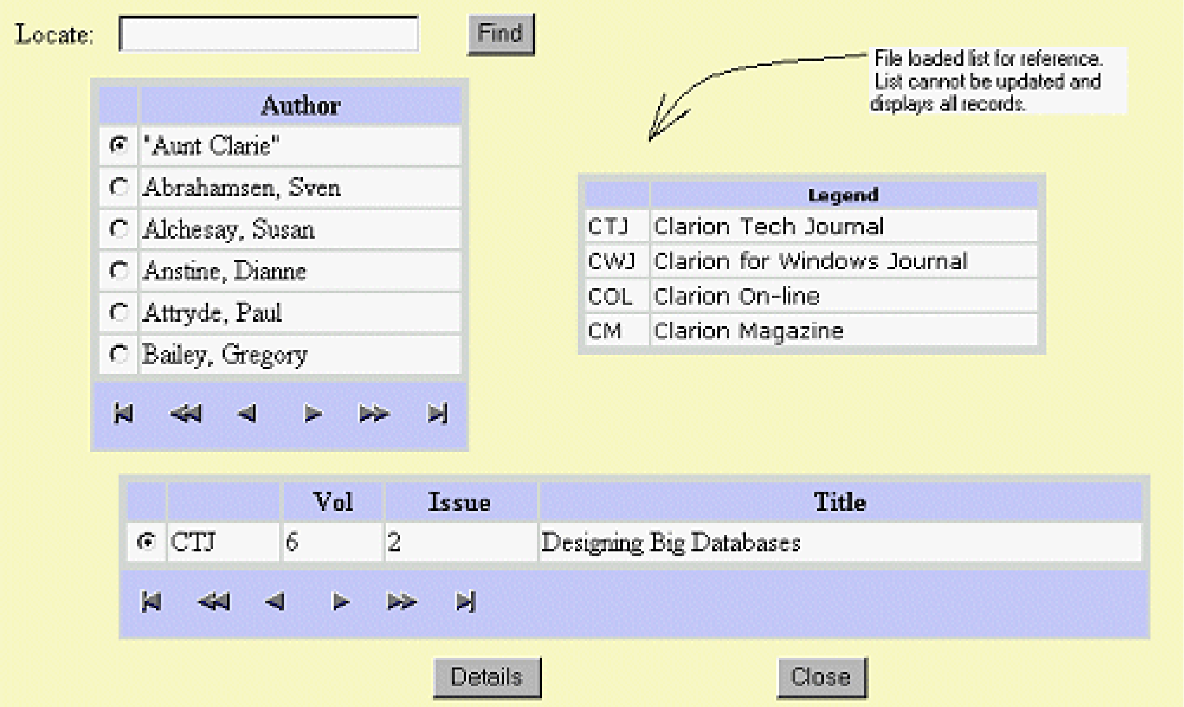


		Volume	Issue	Date	Author / Title	
●	CM	1	11	December 7, 1999	Brewer, Simon	A Nice Place for a Conference
○	CM	1	11	December 7, 1999	Childers, Don	Simple Template Creation
○	CM	1	11	December 7,1999	Stapleton, Andy	SQL Browse runs slow
○	CM	1	11	December 7, 1999	Pickus, Mike	A Question of Style
○	CM	1	11	December 7, 1999	Bayliss, David	ABC Design: The ViewManager (2)
○	CM	1	11	December 14, 1999	Hebenstreit, Tom	ProDomus Translator Plus
○	CM	1	11	December 14, 1999	Harms, Dave	Understanding Clarion Code: Part II

If issue number one is whether I want the skeleton supplied navigation buttons at all and issue number two is where they are placed, issue number three is: Do I have to have the radio (indicator) buttons?

In fact, I do not *always* want them. Suppose I have a small file and I want to display all records (i.e., it’s a small file or I’ll file-load the list). Updates are not allowed and there is no need to select records. In other words, the file data is displayed "FYI," strictly for the user’s information. Figure 3 shows a small file of codes.

Figure 3. Displaying only some tables with navigation buttons.



File loaded list for reference. List cannot be updated and displays all records.

	Author
●	"Aunt Clarie"
○	Abrahamsen, Sven
○	Alchesay, Susan
○	Anstine, Dianne
○	Attryde, Paul
○	Bailey, Gregory

	Legend
CTJ	Clarion Tech Journal
CWJ	Clarion for Windows Journal
COL	Clarion On-line
CM	Clarion Magazine

		Vol	Issue	Title
●	CTJ	6	2	Designing Big Databases

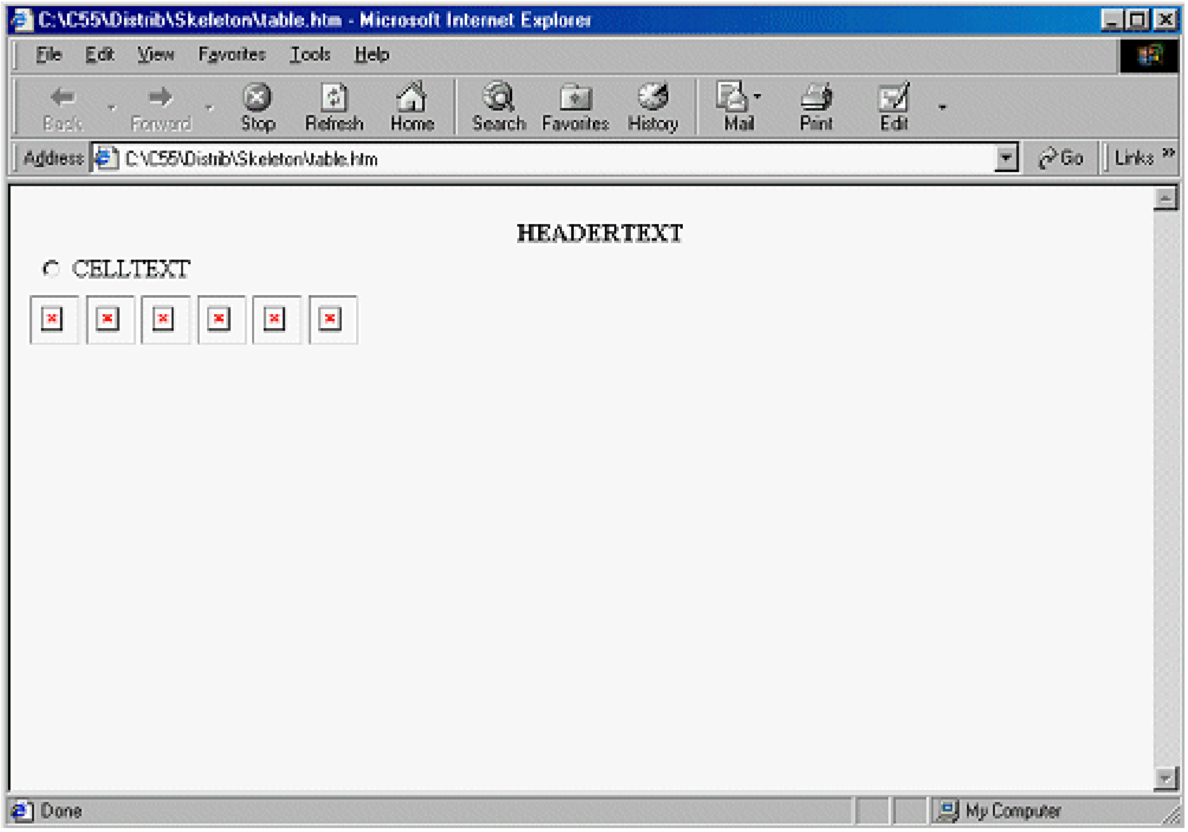
How Lists are Produced

The key to all the issues I’ve just raised lays in how list boxes are rendered into HTML. In other words, knowing how the skeletons structure a standard list box allows making any of the customizations mentioned and allows it much more easily than one might initially expect.

So, how does a list box get presented in HTML?

List boxes are created and controlled by TABLE.HTM. Open TABLE.HTM in a browser and you’ll see something like Figure 4.

Figure 4. TABLE.HTM in a web browser.


C:\C55\Distrib\Skeleton\table.htm - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit
Address C:\C55\Distrib\Skeleton\table.htm
Go Links
Done
My Computer

HEADERTEXT

○ CELLTEXT

This is not actually very revealing. The page shows "HEADERTEXT," which, it seems safe to infer, is likely to be replaced with the column header, the radio/indicator button. It also shows "CELLTEXT," the last being likely to be replaced with field or variable data. Finally, there are spaces where the navigation buttons will go.

Much more information is available with a simple technique dating back to the early days of Internet Connect. Open TABLE.HTM in a text editor and replace each

```
border="0"
```

with

```
border="1"
```

There should be two instances of this which are not in an `<img>` reference. Both are inside `<TABLE>` tags. Therefore, the `tables` will display with visible borders.

Figure 5. The table with cells revealed.

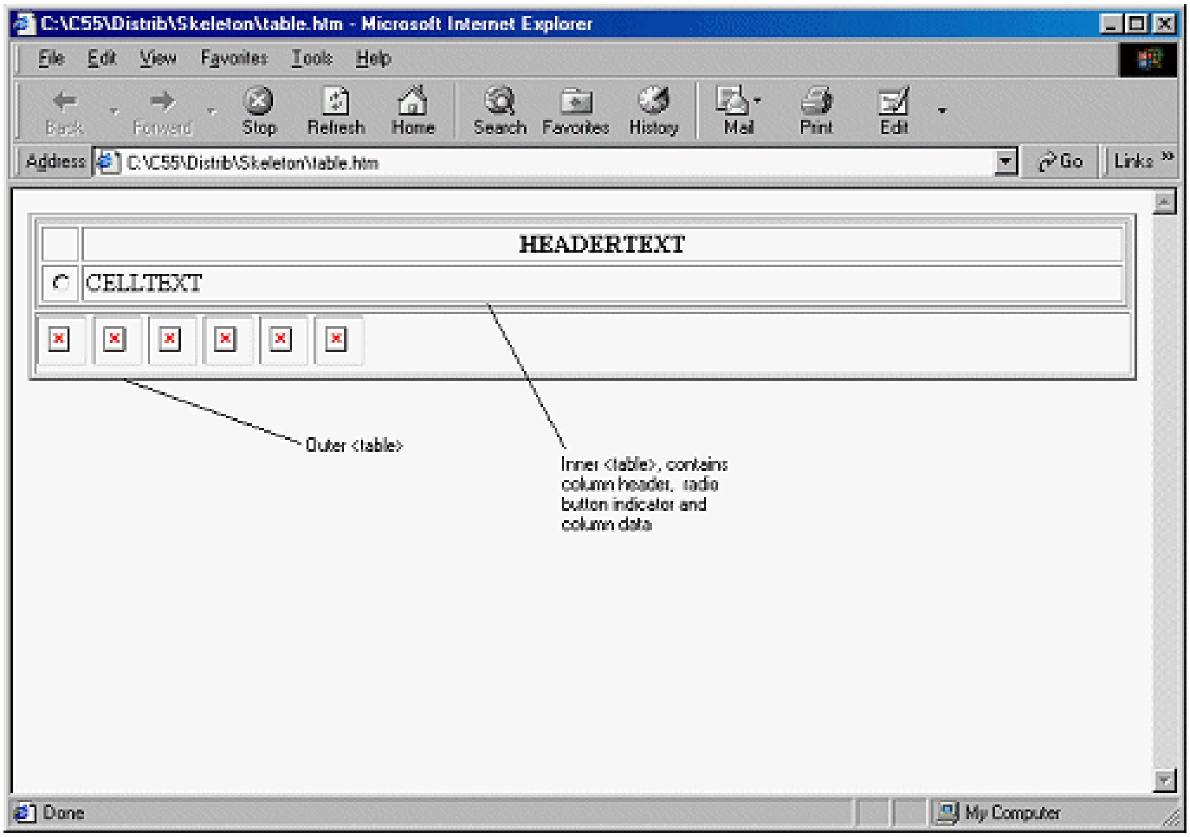

C:\C55\Distrib\Skeleton\table.htm - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit
Address C:\C55\Distrib\Skeleton\table.htm
Go Links
HEADERTEXT
CELLTEXT
Outer <table>
Inner <table>, contains column header, radio button indicator and column data
Done
My Computer

Figure 5 is quite a bit more informative. It demonstrates that the underlying structure is a `<TABLE>` within a `<TABLE>`. The inner `<TABLE>` contains the list box area. The outer `<TABLE>` also contains a row for the navigation buttons. That is, the list box, contained in its own `<TABLE>,` is on one row of the master `<TABLE>` and the navigation buttons are on another row.

From this alone, how to relocate the navigation buttons is obvious: open up the outer <TABLE> (a pry bar should work nicely), move the cell containing the navigation buttons to the desired location and resize. Simple.

Right. (Patience, I’ll come back to this.)

The next obvious question is this: The structural picture of the HTML shows a single row for the column headers and a single row/cell for the field data; how are multiple columns and rows written into the final HTML? This is actually a bit of a digression, as it does not directly impact the skeleton customizations I want to do, but it *is* very intriguing. It is also very informative.

The actual code in the skeleton is as follows:

```
<tr FinalColor=HeaderB>
  <th width="2">
    &nbsp
  </th>
  <TSSCRIPT repeat times="FromColumns" name=column>
  <th>
    <TSSCRIPT value="ColumnHeader[column-1]">
    HEADERTEXT !Column Header
    </TSSCRIPT>
```

Later there is similar code for the radio button indicator and field data. (In TABLE.HTM, search for either "repeat" or for the HTML literals shown in Figures 4 and 5.)

Clearly, TSSCRIPT passes the required number of iterations, horizontally and vertically to the HTML parser. The result is the production of the correct number of column heads (horizontal cells) and the same number of (vertical) rows as appear in the list box when it is running in Windows.

In fact, if you examine each `<TSSCRIPT> </TSSCRIPT>` block in isolation, you will notice that each writes a *single* cell of information. Moreover, each very much resembles a `LOOP` structure surprisingly like what you or I would write were we coding this by hand. There is a `repeat` that writes the headers. Then there is another that writes the rows, placing a radio button at the beginning of the row and then another `repeat` for the fields. Finally, the `repeat` for the rows terminates. (If you go back to Clarion (DOS) 1.x, this ought to sound remarkably familiar, right down to the "repeat structure" terminology.)

So the secret is that list boxes are produced one line at a time by nested loops.

First, an HTML `<TABLE>` is initialized (i.e., one line is written). Then a second HTML `<TABLE>` is started.

In this second `<TABLE>`, cells (`<TD></TD>` pairs) containing the text of all the needed column headers are written. Then cells for the data fields are written and the second `<TABLE>` is closed.

Finally, a row containing the navigation buttons is written and the first `<TABLE>` is closed.

So, now I know the code I'm about to muck about with.

Removing List Box Navigation Buttons

(First, a bit of advance information: I have been informed that, later, there will be a browse-level toolbar template. This template will be WebBuilder-specific. If it is present, the navigation buttons supplied by the skeletons will not *be generated.)*

Given the structures created by TABLE.HTM, removing the skeleton-supplied navigation buttons becomes a simple matter of removing (or, my preference, commenting out) the entire `<TABLE>` row containing them:

```
<!-- hide
<TSSCRIPT include="NavigationControls">
<tr>
  <td FinalColor=HeaderB>
    <TSSCRIPT tag=a attr=href replace="NAME" value="Name">

!Button definitions and scripts go here </TSSCRIPT> </td> </tr>
</TSSCRIPT> -- >
```

And, yes, it *is* literally that simple.

For those who like to read OO code, there is a method call that can be used (such persons, of course, don't even dare fantasize about membership in the Lazy Programmer's Society).

In `Web:Browse:1.GetProperty` (where "`Browse:1`" is the Use variable of the browse control for which skeleton-generated navigation buttons are *not* wanted), inserting code before the Parent call such as the following:

```
If (name = 'NavigationControls')
  Return CreateBoolValue(False)
End
```

will prevent the extra set of navigation buttons from being generated.

Here is the decision the developer must make: to code or to skeleton-ize.

If you always use toolbars, then customizing TABLE.HTM means that you *never* have to be concerned with the extra set of buttons. Since the `GetProperty` call would have to be embedded for *each* list box, it is clear that a modified TABLE.HTM is the way to go.

On the other hand, if you never use toolbars, you probably do want to allow end users to "scroll." That means that you will want the skeleton navigation buttons to generate for all but a few, exceptional, browses. For the occasional list where navigation is unnecessary, the legend box in Figure 3, for example, it is almost a toss up. However, it is my thinking that nominating a custom skeleton is less work than locating the correct embed and typing the code (much less remembering it); the custom skeleton is still the way I’d go.

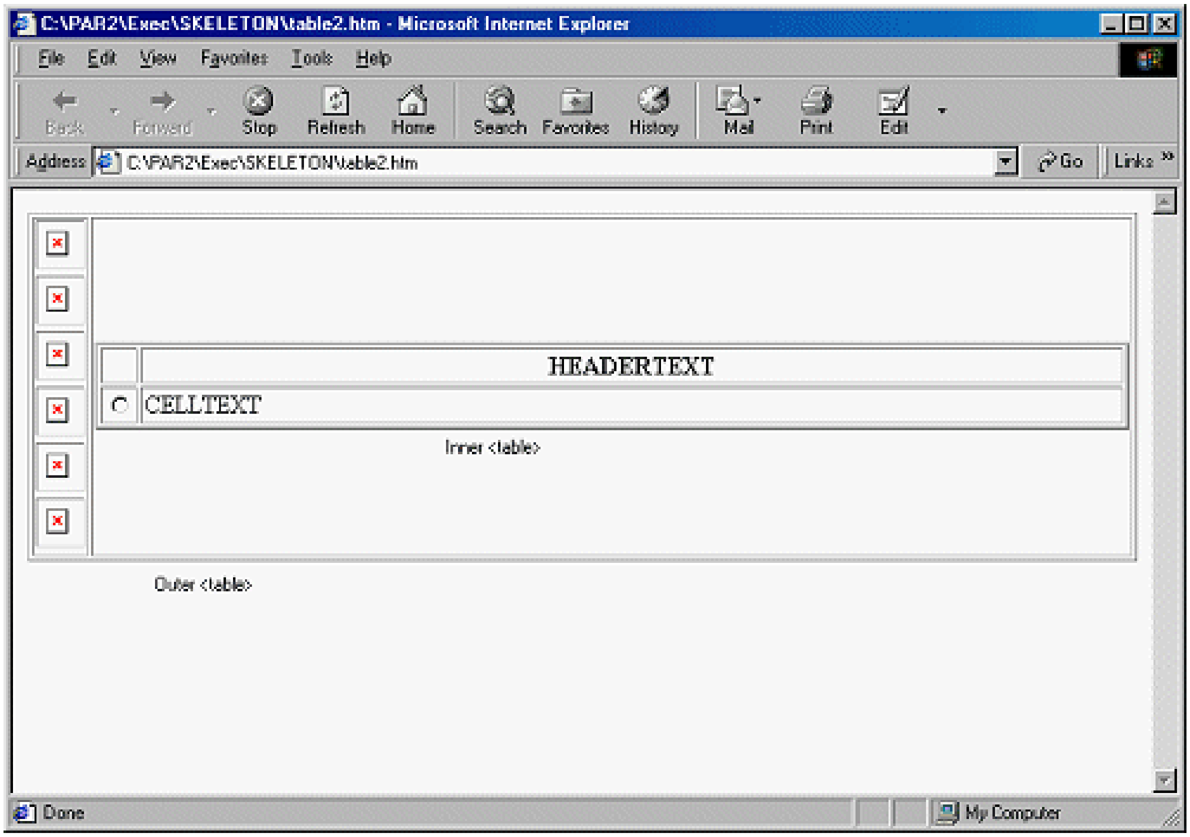
In the case illustrated by Figure 3, a custom skeleton is really the only way to go. While I can prevent the generation of the navigation buttons in code, I cannot prevent the production of the indicator buttons in a similar fashion.

Relocating The Navigation Buttons

Figure 2 shows the navigation buttons on the left of the list area. So obviously this can be done. And it is done in *just* the manner mentioned earlier. To accomplish this re-placement of the buttons, I moved the entire cell containing the navigation buttons so that it came before the `<TABLE>` containing the list, which essentially reverses their placement in the outer `<TABLE>`. A few minutes with a visual HTML editor cursing … I mean, manipulating the size and placement did it.

Graphically, the skeleton looks like Figure 6.

Figure 6. The modified skeleton (with borders showing).



Simple.

(TABLE2.HTM, the skeleton used to produce the browse in Figure 2 is available for download.)

An important modification would be the use of different images. Left and right pointing arrows really do look funny here; at least they do to me. For a production app, I would use a set of up and down pointing pictures.

Of course, you can also center the header text and change the font.

Getting Rid Of The Radio Buttons Too

The final issue I raised initially was: Do I really have to have the radio (indicator) buttons?

If the radio button indicator is not wanted, removing it is almost as simple as removing the navigation buttons.

The obvious thing to do is delete or comment out the `TSSCRIPT` which creates them:

```
<TSSCRIPT tag=input attr=onClick text="icSubmitForm()" ↵
  when="SubmitOnChange">
  <input type="radio" value="ROW" name="NAME$Choice"↵
    id="FEQ$ROWNO">
</TSSCRIPT>
```

Doing so, however, still leaves the cell in which they were displayed visible. Granted, it is not large but even this small, empty cell is unprofessional looking.

In fact, the entire cell, beginning with

```
  <td width="2">
```

several lines above, to the closing `</td>` tag must be removed, a total of 15 lines.

Summary

This is the fourth article in this series but it is the first in which the solution to a problem was arrived at by understanding the underlying template (a skeleton really is a template, isn't it?) rather than trial-and-error. Perhaps it's just me.

In any case, it seems that as I dig deeper into the application of WebBuilder technology, the more systematic it gets. This is a very good thing.

Still to come: relocating list boxes and the skeleton bidding system.

Steve Parker started his professional life as a Philosopher but now tries to imitate a Clarion developer. A former SCCA competitor, he has been known to adjust other competitor's right side mirrors - *while on the track (but only while accelerating). Steve has been writing on Clarion since 1993.*

Main Menu | Log In | Subscribe | Links | FAQ | Advertising

Clarion MAGAZINE

Clarion Development Resources

published by CoveComm Inc.

Main Page

Log In
Subscribe
Renewals

Frequently Asked Questions

Site Index
Article Index
Author Index
Links To Other Sites

Downloads
Open Source Project
Issues in PDF Format
Free Software

Advertising

Contact Us

Search

Feature Article

February, 2000

Skeleton Basics V: Listbox Relocation
(Feb 15,2000)

Edit-In-Place:Lights, Camera, Action...Take Two, Part Two!
(Feb 15,2000)

Skeleton Basics Part 1

Skeleton Basics Part 2

Skeleton Basics Part 3

Skeleton Basics Part 4

Skeleton Basics Part 5

Skeleton Basics V: Listbox Relocation

by Steve Parker

When the World Wide Wait first became popular, 28.8 modems were cutting edge and we were, by and large, using VGA (640 x 480). CAD/CAM designers, engineers and the wealthy had SVGA monitors. Web designers used every pixel they could and, somewhere between often and usually, a few more, quite a few more. Now that 1024 x 768 is all but ubiquitous (though some laptops are still limited to SVGA's 800 x 600), the latest style in page design is to restrict the maximum horizontal width to 600 pixels, leaving a nice, wide, empty area at the right.

Go figure.

This style also seems to involve moving lists or text to the right, using only part of the available horizontal real estate. The area freed on the left is given over to graphics (some of which actually convey useful information...or so I have heard).

This means that Clarion developers need to know how to restrict the width of a page set. Knowing how to restrict listboxes and moving them about the page is also necessary, if only to teach the resident web-head some basic skills.

These tasks turn out to be not only fairly straightforward but border on simple, especially when compared to some of the challenges covered in the first four installments in this series. This time, it's almost entirely a matter of applying what has been learned about how WebBuilder constructs HTML.

Two facts determine how a WebBuilder page is constructed. First, all page elements are created within an HTML table. The second important fact is that a single skeleton (WinCore.HTM in beta 1; WinCore, ColorA and ColorB will be merged into Window.HTM in future releases) controls the basic page structure.

Restricting Page Size

If all controls are in an HTML `table` but `tables` are often nested, the first `table` declaration controls the width of every page created.

In the current implementation, the page width is set to 100% (see Figure 1).

DeveloperPlus
Your source for development tools and add-ons.
Your outlet for application sales.

Figure 1. Default table code for a ClarionForm.

```
<form name="ClarionForm" method="GET" action="PROGRAM.TARGET"
onsubmit="return (submitSuppress-- == 0);"> <input type="HIDDEN"
name="__Special__" value> <table finalcolor="Border" border="0"
cellpadding="4" cellspacing="2" width="100%"> <tr
finalcolor="Header"> <td width=100%><b> <TSSCRIPT value=Title> Page
Title
```

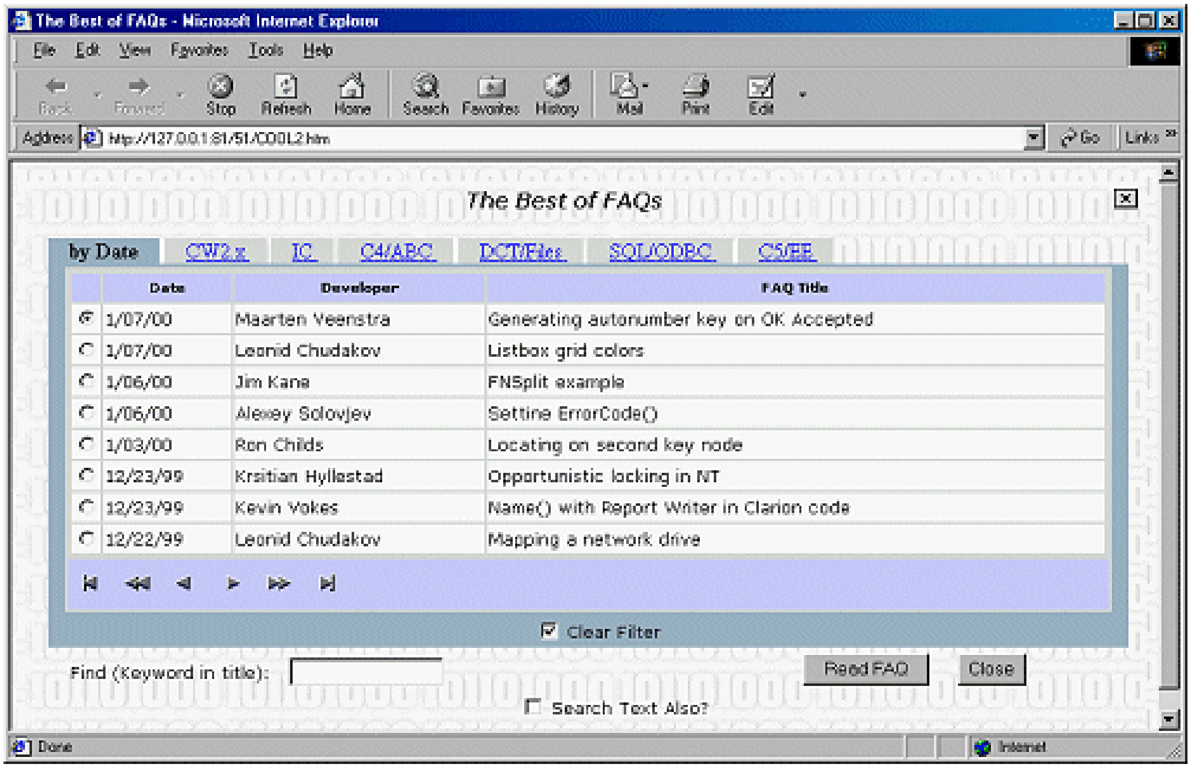
It should, therefore, be a simple matter of changing this from the relative 100%, which uses all horizontal screen space, to an absolute 600 (or 800 or whatever) pixels, as in Figure 2 (the absence of the percent symbol makes the parameter an absolute number of pixels).

Figure 2. Fixing the page width.

```
<table finalcolor="Border" border="0"
   cellpadding="4" cellspacing="2" width="600">
```

Comparing Figure 3 (standard page) with Figure 4 (600 pixels) confirms the result.

Figure 3. The page at 100% width.


The Best of FAQs - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit
Address http://127.0.0.1:81/51/COOL2.htm
Go Links
by Date
CW2.x
IC
C4/ABC
DCT/Files
SQL/ODBC
C5/EE
Read FAQ
Close
Done
Internet

The Best of FAQs

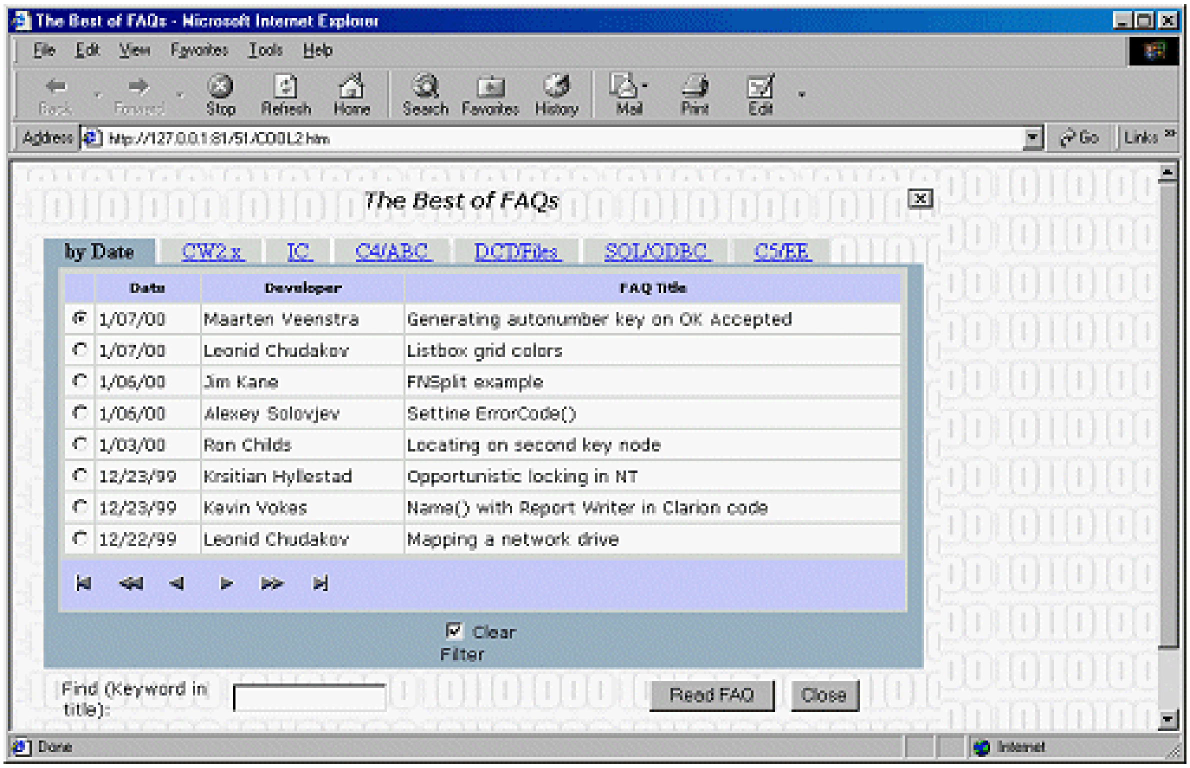
	Date	Developer	FAQ Title
●	1/07/00	Maarten Veenstra	Generating autonumber key on OK Accepted
○	1/07/00	Leonid Chudakov	Listbox grid colors
○	1/06/00	Jim Kane	FNSplit example
○	1/06/00	Alexey Solovjev	Settine ErrorCode()
○	1/03/00	Ron Childs	Locating on second key node
○	12/23/99	Krsitian Hyllestad	Opportunistic locking in NT
○	12/23/99	Kevin Vokes	Name() with Report Writer in Clarion code
○	12/22/99	Leonid Chudakov	Mapping a network drive

☑ Clear Filter

Find (Keyword in title):

□ Search Text Also?

Figure 4. The page at 600 pixels width.


The Best of FAQs - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit
Address http://127.0.0.1:81/51/COOL2.htm
Go Links
The Best of FAQs
by Date
CW2.x
IC
C4/ABC
DCT/Files
SQL/ODBC
CS/EE
Done
Internet

	Date	Developer	FAQ Title
●	1/07/00	Maarten Veenstra	Generating autonumber key on OK Accepted
○	1/07/00	Leonid Chudakov	Listbox grid colors
○	1/06/00	Jim Kane	FNSplit example
○	1/06/00	Alexey Solovjev	Settine ErrorCode()
○	1/03/00	Ron Childs	Locating on second key node
○	12/23/99	Krsitian Hyllestad	Opportunistic locking in NT
○	12/23/99	Kevin Vokes	Name() with Report Writer in Clarion code
○	12/22/99	Leonid Chudakov	Mapping a network drive

☑ Clear Filter

Find (Keyword in title):

> NOTE: Splash screens, toolbars and menus may require modifying other skeletons but the modification would be just the same.

Resizing Listboxes

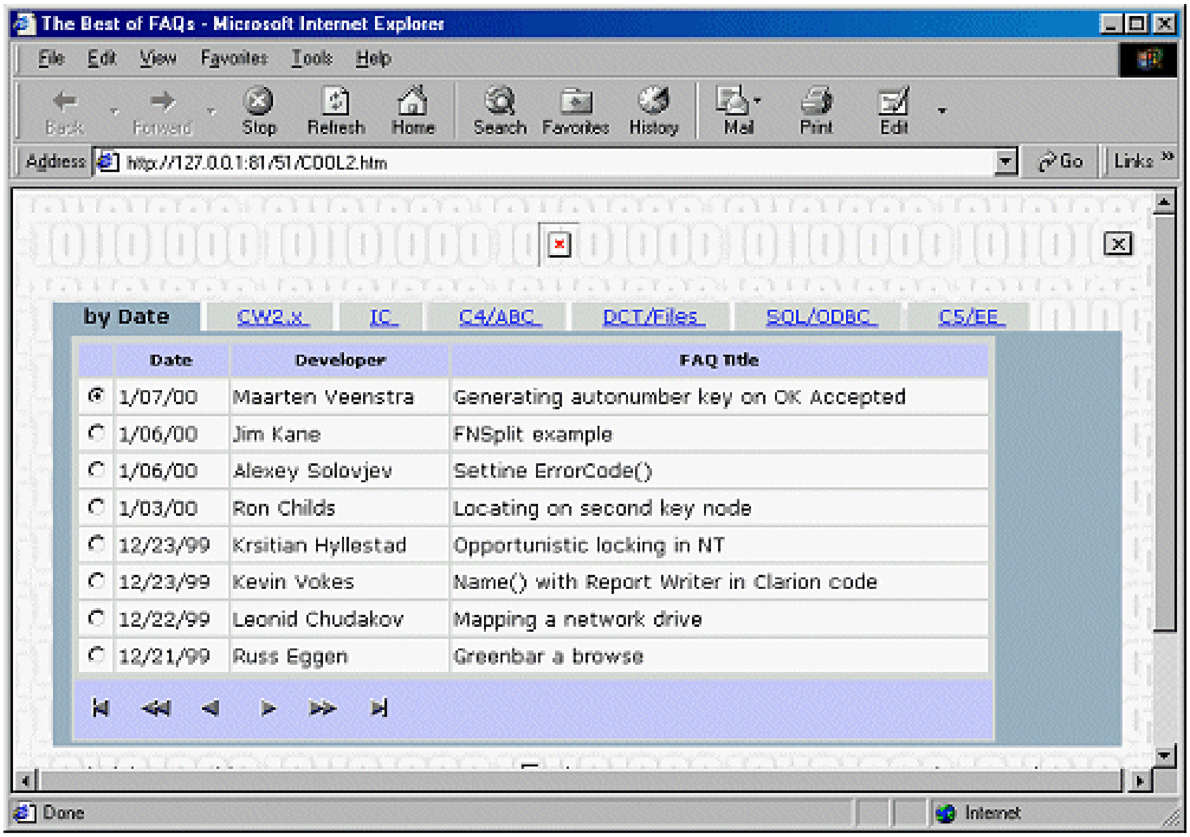
Resizing the listbox – making it smaller and moving it to one side - inside the overall page is a bit harder. It is less straightforward than restricting the overall size of the page because there are several ways to accomplish the goal.

Perhaps the easiest thing is to place a control to one side of the listbox. An image control is an ideal candidate (even on the Web, an image control's contents *can* be set dynamically). Not very interesting, this; the technique has been used since CWIC1 beta 1 (particularly using blank.gif to give some additional space between controls).

> If you are not familiar with this technique: in the window designer, resize the listbox and move it to one side. In the space freed up, place an image control. As long as an image is supplied (even an empty one) but with a fixed size, the list will not expand to fill the available horizontal space. The shortcoming of this technique is only images can be display next to the list. On the other hand, it may be precisely what is needed for a specific application.

Since Table.HTM controls creation of list boxes, it would be logical to restrict the size of the outer <TABLE> in Table.HTM. The result does indeed make the list smaller.

Figure 5. Making the list box smaller.

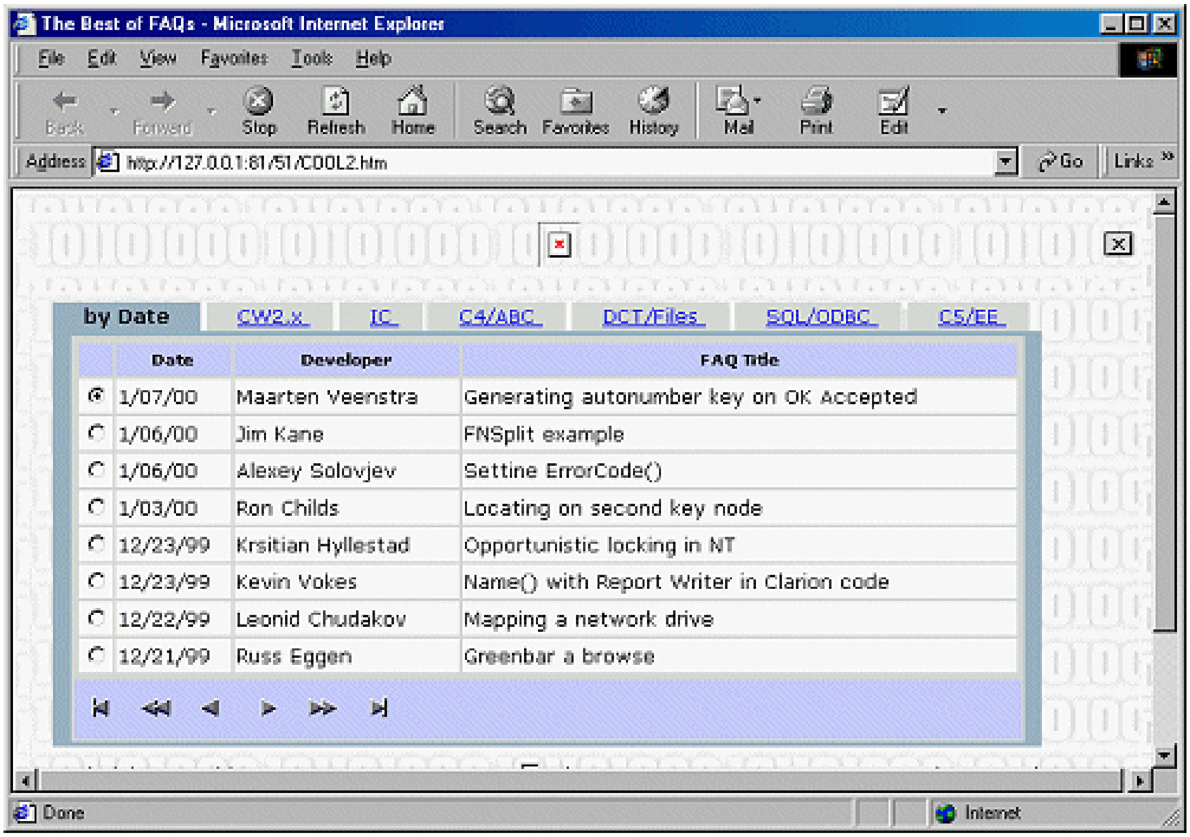

The Best of FAQs - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit
Address http://127.0.0.1:81/51/COOL2.htm
Go Links
by Date
CW2.x
IC
C4/ABC
DCT/Files
SQL/ODBC
C5/EE
Done
Internet

	Date	Developer	FAQ Title
●	1/07/00	Maarten Veenstra	Generating autonumber key on OK Accepted
○	1/06/00	Jim Kane	FNSplit example
○	1/06/00	Alexey Solovjev	Settine ErrorCode()
○	1/03/00	Ron Childs	Locating on second key node
○	12/23/99	Krsitian Hyllestad	Opportunistic locking in NT
○	12/23/99	Kevin Vokes	Name() with Report Writer in Clarion code
○	12/22/99	Leonid Chudakov	Mapping a network drive
○	12/21/99	Russ Eggen	Greenbar a browse

However, most list boxes are on sheet controls and the sheet doesn't appear to be included in the resizing. If the webmaster intends to add a new cell to the sheet skeleton in order to place things in, this will do. Otherwise, one could argue that it looks a bit odd.

Sheet.One.HTM creates multi-tabbed controls. Setting *its* width to an absolute number of pixels does do the job originally visualized:

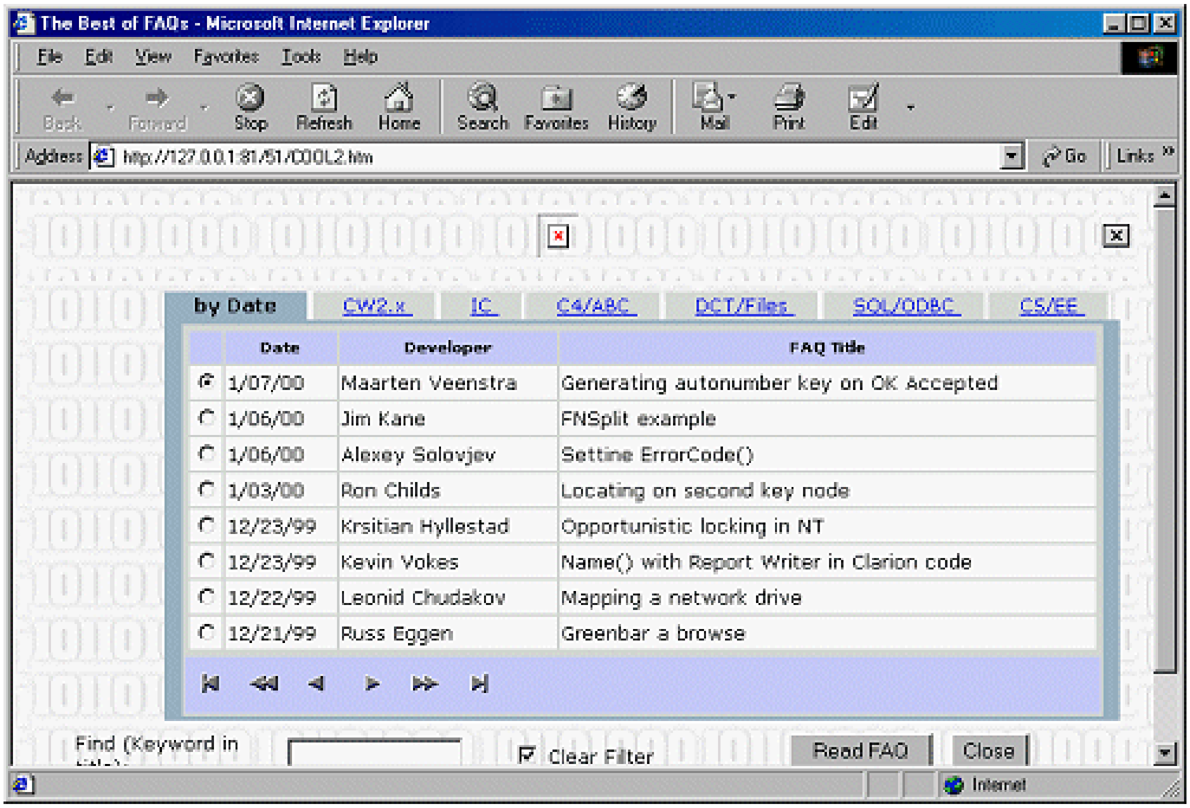
Figure 6. The page with sheet size set.


The Best of FAQs - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit
Address http://127.0.0.1:81/51/COOL2.htm
Go Links
by Date
CW2.x
IC
C4/ABC
DCT/Files
SQL/ODBC
C5/EE
Done
Internet

	Date	Developer	FAQ Title
●	1/07/00	Maarten Veenstra	Generating autonumber key on OK Accepted
○	1/06/00	Jim Kane	FNSplit example
○	1/06/00	Alexey Solovjev	Settine ErrorCode()
○	1/03/00	Ron Childs	Locating on second key node
○	12/23/99	Krsitian Hyllestad	Opportunistic locking in NT
○	12/23/99	Kevin Vokes	Name() with Report Writer in Clarion code
○	12/22/99	Leonid Chudakov	Mapping a network drive
○	12/21/99	Russ Eggen	Greenbar a browse

Add the align="right" (or "center," "left" is the default) attribute to get the desired final look.

Figure 7. Changing the list box alignment.


The Best of FAQs - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit
Address http://127.0.0.1:81/51/COOL2.htm
Go Links
by Date
CW2.x
IC
C4/ABC
DCT/Files
SQL/ODBC
C5/EE
Find (Keyword in
☑ Clear Filter
Read FAQ
Close
Internet

	Date	Developer	FAQ Title
●	1/07/00	Maarten Veenstra	Generating autonumber key on OK Accepted
○	1/06/00	Jim Kane	FNSplit example
○	1/06/00	Alexey Solovjev	Settine ErrorCode()
○	1/03/00	Ron Childs	Locating on second key node
○	12/23/99	Krsitian Hyllestad	Opportunistic locking in NT
○	12/23/99	Kevin Vokes	Name() with Report Writer in Clarion code
○	12/22/99	Leonid Chudakov	Mapping a network drive
○	12/21/99	Russ Eggen	Greenbar a browse

This opens the possibility of adding a new table or new cell for banner ads or other germane or critical corporate information.

Summary

This stuff is beginning to get easy. Well, if not easy, it's beginning to make sense, or follow a method.

Okay, the basics are done. As the great teacher Hillel said, *apropos* a slightly different circumstance, "the rest is commentary." The next clause of the quote is "Now, go forth and learn."

Steve Parker started his professional life as a Philosopher but now tries to imitate a Clarion developer. A former SCCA competitor, he has been known to adjust other competitor's right side mirrors - while on the track (but only while accelerating). Steve has been writing on Clarion since 1993.

Clarion MAGAZINE

Clarion
Development
Resources

published by
CoveComm Inc.

Skeleton Basics V: Listbox Relocation
(Feb 15,2000)

Edit-In-Place:Lights, Camera, Action...Take Two, Part Two!
(Feb 15,2000)

Feature Article

February, 2000

Edit-In-Place:
Lights, Camera, Action ...Take Two!

by James Cooke

Part 2 of 2

In last week’s issue I discussed a dilemma that often faces programmers: Should I use edit-in-place (EIP) or menu-browse-form? The point came out that EIP is fine for users as long as they only have to update a few fields, and it is fine for programmers as long as the logic stays simple. The problem is that EIP causes very complex procedures, not only because of the amalgamation of the browse and form logic, but also the loss of handy control, code and extension templates that ease up validations and other operations. *Users* want EIP, but *programmers* want menu-browse-form!

The idea then, is to "morph" the update form using property syntax to make it *look like* EIP, the main benefit being to keep the original update form with all embeds, templates and controls intact!

Setting Up The Browse Procedure

There is not much to do in the browse procedure; simply make sure that the order and quantity of columns in the browse match the order and quantity of fields in the update form. Since edit in place is cell-oriented, set the columns attribute on the list box to TRUE . To "lock" the list box equate label to a specific value, append the list box equate label with ',1000'.

Setting Up The Update Procedure Variables

Declare these local variables in the update form as follows:

Figure 1. Local variables required in the update form

```
EditInPlaceVariables GROUP,PRE(EIP)
!number to compensate for incremental font creep
FontSizeCompensator  SHORT
!the cell the user clicked on
CurrentColumn        BYTE
!total number of columns in the listbox
NumberColumns        BYTE
!width of the listbox (needed for positioning form)
ListWidth            USHORT
!width of the listbox (needed for positioning form)
ListboxPos           SHORT
!Xpos of listbox, relative to entire screen
InsertPos            SHORT
!Xpos of the clicked on cell, relative to entire screen
RowPos               SHORT
!Height in pixels of the highlighted row
RowHeight            BYTE
!Listbox typeface (prop:font,1)
Font1                STRING(20)
!Listbox font size (prop:font,2)
Font2                BYTE
!Listbox font color (prop:font,3)
Font3                LONG
!Listbox font style (prop:font,4)
Font4                LONG
!Counter for internal operations
ControlNum           SHORT
!Accumulator for internal operations
AccumPos             SHORT
!Counter for internal operations
Column               SHORT
!Picture of current cell
ColumnPicture        CSTRING(20)
!Width of current cell
ColumnWidth          BYTE
                     END
!Queue of column attributes
qEIPColumns          QUEUE
!Picture of column
ColumnPicture        CSTRING(20)
!Width of column
ColumnWidth          BYTE
                     End
```

These variables must be assigned before the update form's window opens.

A good place to put the following code is after the embed "Set options from global values", priority 5400. Bear in mind that *all control property assignment statements at this point refer to the browse window, even though the code resides in the update form.*It is important to ensure that both units of measurement are in pixels and not dialog units before any assignments can take place. Type this code immediately before any property assignments:

```
0{prop:pixels}=1
```

The code in Figure 2 reads the properties of the browse window and assigns the values to local variables.

Figure 2. Priming the update form's local variables

```
0{prop:pixels}=1
EIP:CurrentColumn = 1000{Prop:Column} ! highlighted column
EIP:RowHeight = 1000{prop:LineHeight} !Height of listbox row
EIP:RowPos = 1000{prop:Ypos} +| !Form position on update
  0{prop:ypos} + |
  1000{prop:HeaderHeight} +|
  1000{prop:LineHeight}+|
  (1000{prop:selected,1} * |
  1000{Prop:LineHeight}) - 2
EIP:InsertPos = (1000{prop:Ypos} +| !Form position on insert
  0{prop:ypos} + |
  1000{prop:HeaderHeight} +|
  1000{prop:LineHeight}*2) - 2
EIP:ListboxPos      =   1000{prop:Xpos}+0{prop:xpos}+3
EIP:ListWidth       =   1000{prop:width}
EIP:Font1           =   1000{prop:font,1}
EIP:Font2           =   1000{prop:font,2}
EIP:Font3           =   1000{prop:font,3}
EIP:Font4           =   1000{prop:font,4}
SELF.FirstField     =  |
   choose(self.request=InsertRecord,1,EIP:CurrentColumn)
clear(qColumns)
LOOP EIP:Column = 1 to 4 !Number of browse columns
   qColumns.ColumnPicture  =   |
     1000{PROPLIST:picture,EIP:Column}
   qColumns.ColumnWidth    =   |
     1000{PROPLIST:width,EIP:Column} +1
   add(qColumns)
END
```

Setting The Update Procedure Attributes

There are several conditions in the update form that need to be set before coding begins. They are as follows:

- Remove any extension templates that relate to the resizing of controls
- Set the Frame type to None
- Set Immediate flag to FALSE
- Set 3D Look to FALSE
- Turn off Window Behavior's "Save and Restore Window Location"
- Remove all sheets and tabs, but keep the entry, spin, droplist and check controls with their prompts
- Remove the Cancel Button
- The tab order should ideally be from the top to the bottom of the form.
- Remove the window caption

- Notice that in Listing 3 there is no caption on the window - it will not be needed in an EIP situation. However, even though it is removed, ABC assigns to the window some text at a certain point - the standard "Record will be added/Changed/Deleted" messages. The following code will make sure that piece of ABC code does not execute: Type this at the embed point ThisWindow.Ask(), Before Parent Call, Priority 3800:

  ```
  0{Prop:Text}=''
  ```

Figure 3 is an example of an update form with the entry controls first and the prompts last. The prompts are destroyed at run time.

Figure 3. Vanilla example of "ideal" layout of an update form.

```
Window WINDOW,AT(,,260,100),MDI,SYSTEM,NOFRAME
       ENTRY(@s20),AT(115,8,60,10),USE(CUS:Name),CAP
       ENTRY(@d6),AT(115,24,60,10),USE(CUS:Date)
       ENTRY(@s2),AT(115,40,60,10),USE(CUS:State)
       PROMPT('Customer'),AT(59,10),USE(?CUS:Name:Prompt)
       PROMPT('Date:'),AT(55,25),USE(?CUS:Date:Prompt)
       PROMPT('State:'),AT(55,42),USE(?CUS:State:Prompt)
     END
```

Morphing The Update Form

After opening the window, the first thing to do is to set the unit of measurement for the local screen to pixels and loop through the update form's controls, destroying all unwanted controls and hiding any buttons:

Figure 4. Removing all irrelevant update form controls at runtime.

```
0{prop:pixels}=1
Loop EIP:ControlNum=FirstField() to LastField()
   If EIP:ControlNum{prop:type}<>create:entry  and |
      EIP:ControlNum{prop:type}<>create:button and |
      EIP:ControlNum{prop:type}<>create:list   and |
      EIP:ControlNum{prop:type}<>create:spin   and |
      EIP:ControlNum{prop:type}<>create:check  and |
      EIP:ControlNum{prop:type}<>create:option
      destroy(EIP:ControlNum)
   END
   If EIP:ControlNum{prop:type}=create:button |
      then hide(EIP:ControlNum).
END
```

Then use the belt and braces approach to make sure the attributes of the update form conform to EIP requirements, as shown in Figure 5.

Figure 5. Setting the attributes of the update form at runtime.

```
0{prop:double}=0
0{prop:resize}=0
0{prop:text}=''
0{prop:status}=0
0{prop:gray}=0
0{prop:column}=1
```

If adding a record, the first column would usually be selected, otherwise it would should be whatever cell the user clicked on in the update form:

Figure 6. Selecting the correct column depending on update mode.

```
If Self.Request    =   InsertRecord
    0{prop:YPos}   =   EIP:InsertPos
else
    0{prop:YPos}   =   EIP:RowPos
end
```

Resize this window to the width of the list box, the height to the height of the list box scroll bar and the horizontal position of the window to the position to that of the list box:

Figure 7. Setting size and position of the update form.

```
0{prop:width} = EIP:ListWidth
0{prop:height} = EIP:RowHeight
0{prop:XPos} = EIP:ListboxPos
```

The fields are shifted around next to each other and resized to match their respective columns. The font size is marginally reduced and compensation is made for the width of the column separators. Because each column may have a different picture, the picture of each column is interrogated separately and applied to the field (but not to checkboxes!). A factor is used to compensate for the effect of the font size on the height of the list box row.

Figure 8. Setting the size and positions of the controls on the update form.

```
EIP:ControlNum=0
EIP:AccumPos = 0
EIP:Column = 0
Loop EIP:ControlNum=FirstField() to LastField()
   EIP:Column += 1
   EIP:ControlNum{prop:height} =   EIP:RowHeight
   EIP:ControlNum{prop:ypos}   =   0
   EIP:ControlNum{prop:font,1} =   EIP:Font1
   EIP:ControlNum{prop:font,2} =   EIP:Font2 - 1
   EIP:ControlNum{prop:font,3} =   EIP:Font3
   EIP:ControlNum{prop:font,4} =   EIP:Font4
   Get(qColumns,EIP:Column)
   EIP:ColumnPicture = qColumns.ColumnPicture
   EIP:ColumnWidth   = qColumns.ColumnWidth
   If EIP:ControlNum{prop:type}<>create:check
      EIP:ControlNum{prop:Text}=EIP:ColumnPicture
   END
   EIP:ControlNum{prop:width}=EIP:ColumnWidth
   sum#+=EIP:ColumnWidth + 2
   EIP:ControlNum{prop:Xpos}=EIP:AccumPos
   EIP:AccumPos += EIP:ColumnWidth
END
```

That's it! Generate, compile and run!

Shortcomings

If the user has more columns in the list box than can be visibly displayed, scrolling is used. Because at this point I have not determined how to calculate the horizontal scroll status of the list box, and hence cannot determine the first visible column of the list box, scrolled columns are not supported. Also, screen processing on slower machines may be a bit sluggish because of the controls being moved around, though this is greatly speeded up by setting `SYSTEM{PROP:DeferMove}` to `TRUE`.

Another more serious issue is that if the user changes the metrics of Windows (system font size etc.), the metrics of the positioning of the update form and controls also goes out, resulting in mis-aligned controls. A possible, albeit complex remedy for this is to make the metrics configurable at run time. If in the example application the controls on the update form(s) do not line up with the columns of the list box, or the update form is not "on top" of the current row, try tweaking the Font Compensation Factor in the template by either increasing it or decreasing it.

Clicking elsewhere on the list box during an edit does not produce the desirable effect of canceling or confirming an update, because the update form is not in a separate thread to the browse. To accomplish that may require something a bit deeper!

The Example Application

Thanks to the simplicity of this method, the code can also be converted to a template. Such a template has been included and is used in the example application. Don't forget to register the template before opening the example app.

The example app demonstrates the use of this EIP technique using a one-to-many relationship between the State and Customer tables. The Customer table has four control types:

- Spinbox - a standard spin with the date in it. On adding a new record, the date is primed to today()
- Droplist - a hard coded droplist
- File Loaded droplist - a lookup to the state table
- Standard entry - a standard entry box with capitalization turned on

There are two update procedures for Customer in this application. The first is an implementation of the template, the second is a hand coded version. The hand coded version is to enable easier understanding of the code, and the template is to enable you to use this technique in your applications.

Implementing The Template

- Follow the steps in the section "Setting up the browse procedure"

- Follow the steps in the section "Setting up the update procedure attributes"
- Add the extension template FormToEIP to the update form
- Fill the prompts with the respective values. The "Font compensation factor" prompt is to allow for the size of the font when positioning the update form over the correct row of the list box. The height of the list box row varies depending on the font, size and style, and at the time of writing this article I have not been able to determine a linear relationship between the font size and height of the list box row. For a standard wizard generated app using standard windows setup, the value of 2 works fine.
- Go!

Conclusion

The use of properties and references to structures is often very under-appreciated by Clarion programmers. This simple example shows how much can be achieved by stretching the property syntax of Clarion and using a bit of imagination. Due credit goes to Andrew Finn, the Cape Town User Group and numerous members of the newsgroups who have given me first class answers to my questions on properties, language and templates. Without this fantastic support structure in place, a lot of my programming would just *not work*!

Download the source

James Cooke is currently using Clarion and Informix to develop client server solutions for the fruit export industry. He started in programming in 1991 by writing a CPD POS app for his "hardware store" on a fleamarket. He sold it to a hardware shop, and from there was sucked into the IT industry by developing a plethora of apps for several small companies. He spends his time reading, listening to classical music, scuba diving and hiking. He lives in Cape Town, South Africa with his wife and "zoo."

Clarion MAGAZINE

Clarion Development Resources

published by CoveComm Inc.

Main Page

Log In
Subscribe
Renewals

Frequently Asked Questions

Site Index
Article Index
Author Index
Links To Other Sites

Downloads
Open Source Project
Issues in PDF Format
Free Software

Advertising

Contact Us

Search

The Clarion Advisor

February, 2000

The Clarion Advisor: Redirection Files Revisited
(Feb 22,2000)

ToolTalk: Needles And Haystacks
(Feb 22,2000)

Are Accounting Objects Possible?
(Feb 22,2000)

EIP Template Updated
(Feb 22,2000)

The Clarion Advisor: Rethinking Redirection Files

by John Power

I do believe a sense of organization is essential when using computers. Accordingly I have tried to organize all my data under a number of fixed directories. This means I can have one backup routine which does not need changing when new applications or data files are created.

```
C:\AAClarion5
             \Application 1
             \Application 2
C:\AAAccounts
C:\ZTemp
C:\ZClar5Temp
              \Application 1
                        \ClwSource
                        \Obj16
                         \Obj32
```

All important data files are in directories at the top and all unwanted files are in directories starting with 'Z' at the bottom.

Today data files are being scattered far and wide across our hard disks. As Clarion developers are we in control of our data or do we need redirection? The redirection file supplied by Clarion places all created files in your application's directory but when you create a release version or a debug version some of the created files are now placed in subdirectories of the Clarion directory. In the Clarion 5 directory I have:

```
C:\Clarion5\Bin
               \Obj\Release
               \Obj32\Release
```

It drives me up the wall when files are saved in or under a software directory. There is always the possibility of deleting files or directories that are critical to the operation of the software.

DeveloperPLUS
Your source for development tools and add-ons.
Your outlet for application sales.

I decided to let all created files be stored in subdirectories of the application by changing the %ROOT% to a '.' . I was also interested to see the code the new ABC OOP templates produced, so I add two lines in the common section for *.clw and *.inc files, as follows:

```
[Debug16]
*.obj = .\obj
*.res = .\obj
*.rsc = .\obj
*.dbd = .\obj

[Release16]
*.obj = .\obj\release
*.res = .\obj\release
*.rsc = .\obj\release

[Debug32]
*.obj = .\obj32
*.res = .\obj32
*.rsc = .\obj32

[Release32]
*.obj = .\obj32\release
*.res = .\obj32\release
*.rsc = .\obj32\release

[Common]
*.dll = .;%ROOT%\bin
*.tp? = %ROOT%\template
*.trf = %ROOT%\template
*.txs = %ROOT%\template
*.stt = %ROOT%\template
*.clw = .\ClwSource
*.inc = .\ClwSource
*.*   = .; %ROOT%\examples; %ROOT%\libsrc;↵
           %ROOT%\images; %ROOT%\template;%ROOT%\convsrc
*.lib = .;%ROOT%\lib
*.obj = .;%ROOT%\lib
*.res = .;%ROOT%\lib
```

Now my application directories were:

```
C:\Application
               \HelpSrc
               \ClwSource
               \Obj16\Release
               \Obj32\Release
```

This was a great improvement; I was in control of my files. The application directory was neat and tidy with no unwanted files cluttering it. If I needed to look at the Clarion generated source files they can be found easily in their own directory. Unfortunately, after backing up the Applications directory and subdirectories a few times, I realised that I was saving a mountain of garbage and it was growing fast.

I studied the good book (the Clarion Programmers Guide) and found something on macros. Clarion does not say very much about them or how they can be used. An idea then started to form in my mind. *If* I could create a macro that would contain the application name, say %AppName, that would be defined when the application was opened, I could have in the C5ee.ini file the following:

```
[Redirection Macros]
SaveDir=C:\ZClar5Temp\%AppName
```

I realized I replaced some of the '.' with %SaveDir% in the original file I would then have a Power Redirection file as shown below.

I have also added two lines for *.exe and *.shp in the Release16 and Release32 sections. I can then compile both 16 and 32 bit release programs and the Exe files will be safely in their own directories. I can back up the Application directory with very little garbage. All the files I don't need are under one directory, at the bottom of the list of directories and can be deleted very easily.

For the Debug 16 and Debug 32 Compiles, I have added a line for *.exe to keep the Exe and Clw files together, which avoids problems with the debugger not finding the source.

```
C:\AAClarion5\Application
                           \HelpSrc
                           \Exe16\Release
                           \Exe32\Release

C:\ZClar5Temp\Application
                           \ClwSource
                           \Obj16\Release
                           \Obj32\Release

             \Application2
                           \ClwSource
                           \Obj16\Release
                          \Obj32\Release
```

The complete redirection file:

```
[Debug16]
*.exe = C:\ZClar5Temp\%SaveDir%\ClwSource
*.obj = C:\ZClar5Temp\%SaveDir%\Obj16
*.res = C:\ZClar5Temp\%SaveDir%\Obj16
*.rsc = C:\ZClar5Temp\%SaveDir%\Obj16
*.dbd = C:\ZClar5Temp\%SaveDir%\Obj16

[Release16]
*.exe = .\Exe16\Release
*.shp = .\Exe16\Release
*.obj = C:\ZClar5Temp\%SaveDir%\Obj16\Release
*.res = C:\ZClar5Temp\%SaveDir%\Obj16\Release
*.rsc = C:\ZClar5Temp\%SaveDir%\Obj16\Release
```

```
*.dbd = C:\ZClar5Temp\%SaveDir%\Obj16\Release

[Debug32]
*.exe = C:\ZClar5Temp\%SaveDir%\ClwSource
*.obj = C:\ZClar5Temp\%SaveDir%\Obj32
*.res = C:\ZClar5Temp\%SaveDir%\Obj32
*.rsc = C:\ZClar5Temp\%SaveDir%\Obj32
*.dbd = C:\ZClar5Temp\%SaveDir%\Obj32

[Release32]
*.exe = .\Exe32\Release
*.shp = .\Exe32\Release
*.obj = C:\ZClar5Temp\%SaveDir%\Obj32\Release
*.res = C:\ZClar5Temp\%SaveDir%\Obj32\Release
*.rsc = C:\ZClar5Temp\%SaveDir%\Obj32\Release
*.dbd = C:\ZClar5Temp\%SaveDir%\Obj32\Release

[Common]
*.dll = .;%ROOT%\bin
*.tp? = %ROOT%\template
*.trf = %ROOT%\template
*.txs = %ROOT%\template
*.stt = %ROOT%\template
*.clw = C:\ZClar5Temp\%SaveDir%\ClwSource
*.inc = C:\ZClar5Temp\%SaveDir%\ClwSource
*.*   = .; %ROOT%\examples; %ROOT%\libsrc;%ROOT%\images; ↵
     %ROOT%\template;%ROOT%\convsrc
*.lib = .;%ROOT%\lib
*.obj = .;%ROOT%\lib
*.res = .;%ROOT%\lib
```

Unfortunately I can't get the macro to work, so until I do, I changed C5ee.ini to

```
[Redirection Macros]
SaveDir=C:\ZClar5Temp\ZApp
```

This is in case I forget to do the following: I am copying the redirection file to my application directory. Then change all the %SaveDir% to the name of the application. There is one good point: if I reload Clarion I don't lose my redirection file.

Do you have a tip you'd like other Clarion developers to know about? Send it to advisor@clarionmag.com.

Main Menu | Log In | Subscribe | Links | FAQ | Advertising

Clarion MAGAZINE

Clarion Development Resources

published by CoveComm Inc.

Main Page

Log In
Subscribe
Renewals

Frequently Asked Questions

Site Index
Article Index
Author Index
Links To Other Sites

Downloads
Open Source Project
Issues in PDF Format
Free Software

Advertising

Contact Us

Search

The Clarion Advisor: Redirection Files Revisited
(Feb 22,2000)

ToolTalk: Needles And Haystacks
(Feb 22,2000)

Are Accounting Objects Possible?
(Feb 22,2000)

EIP Template Updated
(Feb 22,2000)

Feature Article

February, 2000

Tool Talk: Needles and Haystacks

by Tom Hebenstreit, Reviews Editor

While the World Wide Web has been called many things, what it really amounts to is the world’s biggest haystack. Sure, it is filled with incredible amounts of useful information, but it can require prodigious effort to wade through the garbage and find the gold (Pia Zadora fan sites, anyone?).

As Clarion users, we just *know* that there must be good stuff out there – the problem is finding it. Thus, the focus of today’s column: locating Clarion-specific information and tools on the web (i.e., finding the needle you need when you need it).

Ok, Sherlock, Start Looking

Here are a few good starting points for your journey of discovery.

Clarion Magazine’s Site Index

First off (shameless plug coming!), be sure to check out our own Clarion magazine links page. For your clicking pleasure, the links have been grouped into fourteen general categories such as Data Entry, Graphics, Security, SQL and ODBC, Consultants and more.

http://www.clarionmag.com/common/links_bycategory.html

TopSpeed Turnpike

Although most Clarion sites will have links to other Clarion related sites, there are a few that have specialized in hooking up to every Clarion link they can find. These sites are a great way to get a "big picture" view of Clarion on the web.

Probably the best one of these sites that I have found so far, this site is maintained by "Turnpike Tommy" (otherwise known as Tom Ruby). Links are updated on a regular basis and the site includes links to general programming and tool pages as well as Clarion-specific links.

DeveloperPLUS
Your source for development tools and add-ons.
Your outlet for application sales.

> **Tip:** Be sure to look for more links on any site you access from here, and you'll quickly find yourself amazed at how many Clarion sites there are.

The site has been moving around due to ISP changes. Currently it can be found at: http://members.tripod.com/~tomruby/turnpike/

TopSpeed Corporation

Although TopSpeed has dropped out of the third party sales market, they still maintain their Accessories program as an advertising tool for third party vendors.

Descriptions of the products and links to vendor sites are provided. At last count, there were about 30 items listed there (not counting TopSpeed products).

http://www.topspeed.com/accessories/accessor.htm

Web Rings

A web ring is a collection of sites with a common theme (in this case, the theme is Clarion). All sites on the ring have buttons or links connecting them to the others (Next, Previous, Index, etc). If, for example, you keep clicking on the Next link, you will travel sequentially around the entire ring of sites, eventually ending up back where you started. To access a list of all sites in a ring, click on the ring Index button or link.

Originated by Troy Sorzano and the lads at IPU, the CWSuperRing is the largest and oldest of the Clarion rings. The current index lists 45 sites. You can start at:

http://www.cwsuperpage.com

Another good ring is Clarion Source, originated by third party vendor Gitano Software (currently listing about 18 sites). You can begin your journey through that ring at their site:

http://www.gitanosoftware.com

Lay Your Money Down

So, if you can't get accessories from TopSpeed anymore, where do you go when you want to buy?

In many cases, you can purchase directly from the vendor (once you have found them, of course). Probably the biggest drawback to that option is that vendors' capabilities for handling sales transactions can vary wildly, ranging from only phone calls and checks to fully secure online credit card sales.

Beyond direct sales, there are three other options:

Mitten Software

A long time developer and distributor of Clarion tools, Mitten distributes several product lines, including RAD Accounting, Allerup Report Designer, the DET/DEF Templates, Software by Ragazzi, Mike Hanson's extensive line of Super templates, and more.

Surprisingly, Mitten doesn't sell directly on the web. They do accept orders via standard email, but I have this thing against sending credit card information in plain text across the Internet. Phone and Fax sales are also offered, and they have a 90-day return guarantee on all products.

http://www.mittensoftware.com

As a side note, Mitten is also one of the few places that still offer add-ons for the older Clarion for DOS products.

ClarionShop

With the demise of TopSpeed Accessories sales, two new sites sprang up to pick up the slack (and the cash, I assume). First up was ClarionShop, sponsored by CapeSoft and NextAge Consulting.

ClarionShop has by far the biggest selection of products (over 70, last time I counted), and offers many extra perks like user ratings, a discussion group and direct links to all of the vendors. Vendors include CapeSoft, NextAge, ProDomus, GAP, Sterling Software, Tinman Development, Linder Software, Paragon Development and quite a few more. Secure online ordering via credit card is supported.

By the way, in the interest of fairness the user ratings require at least ten votes before they are displayed, and the rating is the average of all votes. If you use any of the products sold there, I’d like to encourage you to visit ClarionShop and enter your own impressions (good and bad). We will all benefit from the shared insight into which tools are the most useful.

http://www.clarionshop.com

DeveloperPlus

The second of the new Clarion commerce sites, DeveloperPlus is the brainchild of Lee White of LodeStar Software. While it currently doesn’t have as many choices as ClarionShop, it has also set its sights a bit higher, also offering the same sales service to independent developers for *their* products. In other words, DeveloperPlus isn’t just a place to buy development tools, but a place where you can sell your own products as well.

Secure online ordering is available, and they accept an even wider range of credit cards than ClarionShop.

Current vendors include LodeStar, CoveComm (ClarionMag), GAP, Creative PC Solutions (CPSC), Cowboy Computing Solutions, and Resolute Solutions Corp (that’s a lot of solutions…).

http://www.developerplus.com

There really isn’t much of any overlap between Mitten, ClarionShop and DeveloperPlus, so be sure to check them all out. Chances are you will find what you need at one of those sites.

Other Vendor Sites

A few of the larger Clarion Tool vendors aren’t connected with either of the above sales sites. (By larger, I mean vendors that have multiple products and have been around for a while.)

They include:

Nice Touch Solutions, makers of the Wizard line of products (Query Wizard, Report Wizard, etc.). They can be found at:

http://www.nicetouch.com

Gitano Software, authors of G-Reg, G-Cal, G-Calc, G-Sec, G-Notes and others:

http://www.gitanosoftware.com

Needless to say, there are many more vendors than I have space to list here (my apologies to each and every one of them). Take a look at the Turnpike site I mentioned earlier and you'll probably find the links you're looking for.

There's GOLD In Them Thar Pages

One of the unique things about Clarion is that people can not only share programming tips, they can share a template which embodies and simplifies the use of those tips. While many sites offer some freebies, there are a few sites and packages that deserve special note.

TinTools

The TinTools package is put together by Tinman Development, and surely qualifies as the mother-of-all-freebies. Over the last few years, the good folks at TinMan have been collecting and adding new templates on a regular basis to the point where this is the most comprehensive set out there. TinTools can be found at:

http://www.thetingroup.com

Vince Sorensen's ABC ToolKit/WinAPI Toolkit

For the ABC user, Vince's ABC toolkit is tops when it comes to useful templates for lower level and API features. If you still use the legacy templates, check out the WinAPI Toolkit.

http://www.dlcwest.com/~sorev/topspeed/index.html

> **Note:** The ABC Toolkit has been included in TinTools, so if you download that, you won't need to download the toolkit again. There is usually some lag between Vince's ongoing updates and their inclusion in the TinTools package, though, so you may want to check here for the most current version.

Mike Hanson Free Templates

Author of the commercial Super templates, Mike has some very useful free templates available as well:

http://www.boxsoftdevelopment.com

Brain Food

The following two sites come under the category of Knowledge Bases – general repositories of Clarion-related information and files. Both have been around for quite a while, and contain tons of useful tips, tricks and solutions.

Take some time and explore them; I'm sure you'll find them as useful as I have.

Icetips Knowledge Base: http://www.icetips.com

Steve Parker's CWIC Web: http://www.par2.com

Riding Into The Sunset

That wraps up this edition, and I just *know* there are many good sites that were inadvertently omitted from this list. To help remedy that, please drop me a line here at the magazine and tell me where *your* favorite Clarion sites can be found. I'll be glad to publish them in a future column.

In the meantime, happy hunting!

Main Menu | Log In | Subscribe | Links | FAQ | Advertising

Clarion MAGAZINE

Clarion
Development
Resources

published by
CoveComm Inc.

Main Page

Log In
Subscribe
Renewals

Frequently Asked Questions

Site Index
Article Index
Author Index
Links To Other Sites

Downloads
Open Source Project
Issues in PDF Format
Free Software

Advertising

Contact Us

Search

Feature Article

February, 2000

The Clarion Advisor: Redirection Files Revisited
(Feb 22,2000)

ToolTalk: Needles And Haystacks
(Feb 22,2000)

Are Accounting Objects Possible?
(Feb 22,2000)

EIP Template Updated
(Feb 22,2000)

Are Accounting Objects Possible?

by David Podger

Although I have never written an "accounting object," and never heard of anyone claiming to have done so, I have come to think that an armory of them *could* be written, with a most beneficial effect on Clarion developers’ productivity.

Where did this optimism come from? I suppose it started when I began to wonder why there were things in accounting that had endured for centuries. Why had they never changed? Could their stability be used to anchor accounting software by giving it a stable nucleus? As I worked on the answer to these questions, the earliest beginnings of a specification for accounting objects began to form in my mind.

Putting the spec to the test, I have used it to write and install a real-life accounting app. It includes, within one program, accounts receivable and payable, invoicing, general ledger and bank reconciliation. Bedding down my first crop of users has taken up the second half of 1999. I have written the accounting app in Clarion 5a, using the ABC templates, and will share a small part of its code with you as the story unfolds.

To start at the beginning then, what is it that keeps accounting stable?

Inertia In Accounting Procedures

Accounting is a systematic and complex body of objective knowledge. Like any such richly connected knowledge base, axioms and principles can be found at its centre. Examine any particular process within accounting and you will see that it connects back to these central precepts. One could say, in fact, that the central ideas of accounting are *held in place* by these many dependencies and connections. As a result, the centre of accounting has high inertia, or a high resistance to change.

The many professional accounting bodies which exist around the world can also be thought of as a complex set of connections, connections between people in this case. These societies uphold and promulgate the key principles and central virtues of accounting, and flesh them out with the announcement, from time to time, of changes in their secondary regulations.

Further adding to the high inertia of the central elements of accounting is the fact that in every generation thousands upon thousands of accountants-in-training absorb accounting as a mind-set. For most of them the idea of changing the fundamentals of their profession simply does not arise. The firmly embedded foundations, the entrenched, unquestioned notions which their whole training rests upon are there for good.

The above three kinds of inertia support and reinforce each other. The objective logic of accounting, its public structures within society and its internal mental maps all conspire to hold in place a stable nucleus at the heart of accounting. If this nucleus could be implemented in the form of accounting objects, then that software would inherit its high level of inertia and stability.

For all that, most accounting procedures are regularly changed. Accounting records and reports are regularly adapted to new circumstances and periodically reformed. If there is, nevertheless, a "still centre" within accounting, how to identify it and distinguish it from everything that changes?

Identifying The Nucleus

Here is an attempt at a list of unchanging elements:

1. Accounting is an event (or transaction) recording process
2. It employs redundancy (the double-entry rules)
3. Accounting sums all transactions by various categories
4. Distinct kinds of events are recorded, such as actual and budget events
5. Recent transactions are accessed more than old ones
6. Aggregate data is more acceptable the older the transactions
7. Transactions eventually expire with the passage of time
8. Time is divided into arbitrary accounting periods
9. Accounting sums transactions by such periods

The list is certainly general. Its statements are true for the hand-written account books of past centuries and also true for today’s computer implemented systems. But, the statements in the list are so broad as to seemingly say almost nothing.

The list is also very short. There are not that many things you can say about accounting at this very general level. Try and extend the list yourself and see what I mean.

So, is it a short, useless list - or can something be done with it?

Leaving Out Non-Essentials

Now to apply the list to the development of a specification. Remember, the aim is to design just a nucleus, not a whole system. The term "engine" will refer to the working code of an operational nucleus.

The list’s brevity must mean that very little of what is conventionally thought of as "accounting" can appear in the central engine. How such a stripped down engine can work in practice is a question that comes later, but for now, consider what will have to be left out if such a bare-bones list is strictly followed.

Any organization using an accounting system will have its own particular business structure. In conventional solutions, this structure is directly represented within accounting systems using such entities as departments, ledgers and accounts. But as soon as this is done the core of the solution has become particular rather than general. So common is this approach that the logic of report-writers depends upon ‘breaks’ in such fields to generate sub-totals and grand totals. But clearly the lot must go, and *no representation* of external structure can be allowed within the central data repository of the solution.

Since departments, ledgers and accounts are the very stuff of business, they have to be represented *somewhere* within the system, but this must be where they can be more readily changed. Business structures change far more frequently than the general principles of accounting, so the solution should allow these structural changes to occur gracefully, without a software meltdown.

Another thing that has to go is every kind of master file record that contains multiple balances. No matter how you try, as soon as you introduce such a file into any accounting system you have smuggled in the particular. Consider - how do you decide what total fields to include in such a file? To take a simple example – in a debtors "balance forward" system, there will be multiple total fields for the aging of debt - but these very fields derive from a particular set of business rules that some enterprises will not wish to apply.

But balances cannot be removed from accounting. They are a central part of what it delivers and any engine-based solution will have to provide them. The problem is not with the keeping of totals as such, but with totals that comply with and smuggle in a particular business structure. Separating the one from the other has to be a part of the general solution.

Some More Universals

A quick glance at the pen-and-paper processes that delivered classical accounting shows that accounts always had more than one balance. In fact, the balance of an account depended on what you needed. If you were an accountant extracting a trial balance, you first went through the whole ledger, ruling off each account as at a certain date. Then you transcribed a balance from each ruled-off row. But ruling off an account did not stop your book-keepers from continuing to post new journals to the ledgers. A ledger was often spread across a number of physical books for this very reason. So if a book-keeper was asked for the balance of an account, he or she might give the balance as at the last entry posted, not the one above the last ruled line.

The above observations make for a couple of additions to the list above:

10. Accounts have always had fuzzy, time-dependent balances.
11. Accounting has been a multi-user process from its very beginnings.

These are two more necessary capabilities for the nucleus.

Getting On With The Spec

If there is to be ‘*no representation* of any external structure’ within the nucleus, as noted above, then it follows that the nucleus must have its own internal structure and that something must mediate between this and peripheral processes.

There is a very rough analogy between the engine and the Clarion browse class. The browse class handles any sort of file layout, but is itself completely generalised. Its internal structure looks nothing like the browses a user actually sees on the screen. The analogy is rough because the engine has to look after persistent data kept on files or in tables.

What might the structure of this persistent data look like? Necessarily it will be very simple. The above list of the unchanging elements of accounting will give us one solution at least. Taking them in sequence:

5.1 Accounting is an event (or transaction) recording process

5.2 It employs redundancy (the double-entry rules)

Transactions, then, will consist of two or more postings which, when added together, sum to zero. A Transaction Serial Number (`TSN`) identifies them. The higher this number, the more recent the transaction. All transactions are posted to a set of files (tables) with the collective name of `Pool`.

5.3 Accounting sums all transactions by various categories

The most obvious category is Account. A `Pool` Account Number (`PAN`) is applied to every posting. There is a second category, which goes by a multitude of names in the outside world (department, division, job, profit centre, and so on). These names all define ways of grouping events together to describe different functions carried out by an enterprise. To generalize this requirement, a Function Number (`FUN`) is applied to every posting. Many kinds of hierarchies can be constructed using the basic building blocks of `PAN` and `FUN`. But this must be done outside the engine.

5.4 Distinct kinds of events are recorded, such as actual and budget events

Some accounting systems record only one kind of event - actual. Others distinguish more, such as budgets, commitments, quotations, orders and standing entries, to name a few. These different events never change in kind. One may trigger another, as when a quotation to a customer results in an order being received. But they retain their completely distinct identities. It is these different kinds of events that commonly end up as different balance fields in master file (from now on, please read "file" as "table" if you prefer) records, but multiple balances have no place in the nucleus, for the reasons stated above. Instead the `Pool` shall have different `Streams` of data running into it, each stream recording a different *kind* of event.

All the elements for a unique `Pool` key are now present. They are:

- PAN (ascending)
- Stream (ascending)
- FUN (ascending)
- TSN (descending)

A transaction, that is, will belong to one `Stream` and to one `FUN`. It will consist of a set of postings, each having a different `PAN` but the same `TSN`.

5.5 Recent transactions are accessed more than old ones

5.6 Aggregate data is more acceptable the older the transactions

5.7 Transactions eventually expire with the passage of time

Transactions cannot stay for ever in the `Pool` – they eventually wear out their welcome. And, the older they get, the less interesting they are to the user. There are a number of solutions. The one I have chosen is that of dividing up the `Pool` by year. Thus `P99~00.tps` contains all the postings for the 1999-2000 financial year and `P00~01.tps` all those for the next, and so on. I wish I could think of a deep reason for doing this, but cannot. Binding decisions made for no good reason make me nervous, but so far this one has worked. It has the advantage that postings are only ever added to the `Pool`. Once posted to a `Pool` file they are never changed and never deleted. Their removal from a system is accomplished one whole file at a time.

> 5.8 Time is divided into arbitrary accounting periods

> 5.9 Accounting sums transactions by such periods

The concept of "year" is invariant in all cultures even though a year may be solar or lunar, or a combination of both. When it comes to the accounting periods within a year, however, I have fudged. A completely general solution would allow any method of subdivision but I have, in my working application, fixed on 12 months in a year. The engine *should* implement an internal MonthNum defined in a table of cut-off dates or by a set of rules, with any number of such "months" being allowed in a year. My only concession to generality in this version of the engine has been to allow a financial year to begin on any month.

A simple, orthogonal method of storing balances now suggests itself, given the unique key for the Pool and the arbitrary choice of month as the accounting period. A reduced version of the Pool is needed, also divided into files by year. Thus, R99~00.tps holds all of the 1999-2000 year's balances and so on. Separate totals are kept for each Stream:

- by PAN
- by PAN and Month
- by PAN and Function
- by PAN and Function and Month

Even if month *were* to be generalized this would make no difference to the "reduced" set of files. Reduced, by the way, is a cooking term as in, "reduce the mixture over a slow flame". But wait, there cannot be just *one* balance per account (PAN). There is the next guiding principle:

> 5.10 Accounts have always had fuzzy, time-dependent balances.

The solution adopted is never to *change* the balance records in the Reduced files. Instead, but only *add* records when posting. This idea needs some explanation.

Recall the TSN, a unique transaction number for each posting set constituting one transaction. This is implemented as a ULONG (similarly, PAN and FUN are also ULONG). Transactions are in practice added to the Pool a batch at a time. Batches are identified by a date/time stamp. As an aside, the design deals with real-time postings by adding them to a small Ladle file (and adding this to Pool and Reduced when convenient, as a batch). The design requires procedures returning a real-time balance to add anything in the Ladle file to what they find in Reduced. The rest of this explanation can therefore ignore real-time issues.

Every batch has a "highest TSN", that is, the highest Transaction Serial Number to be found within it. Instead of *changing* balance values in Reduced, the updating process *adds* records that are the sums *for one batch*:

- by PAN and highest TSN
- by PAN and Month and highest TSN
- by PAN and Function and highest TSN
- by PAN and Function and Month and highest TSN

Any procedure that requests balances from Reduced begins by obtaining a limiting TSN value (this value is updated in a Control file after each batch is posted). The limiting

TSN value is then passed to every retrieval procedure in the engine as an input parameter. All engine procedures that look up the Reduced file use a SET/NEXT loop (rather than a GET) to fetch balances. A BREAK ends this loop when the limiting TSN is reached.

Remember rule 5.11:

> 5.11 Accounting has been a "multi-user" process from its very beginnings.

As you can see from the above, the engine implements "fuzzy" balances. A long report process can run in parallel with data entry because it can only "see" Pool and Reduced in the "state" they were in when it began running. The solution is inherently multi-user, but requires no use of LOGOUT or ROLLBACK methods. It adds some inefficiency to the fetching of balances, but this is more than offset by the benefits flowing from its robust simplicity.

The same is true for processes that access records from the Pool. None are allowed to fish there directly but must use engine procedures. As a result, they can only see transactions up to and including the limiting TSN they are given when they began.

On a daily basis or more frequently, when the system is not in use, the Reduced file is, ah, further reduced . In a fully reduced state its TSN fields are all zero and there is only one balance for each possible combination of Stream, PAN, FUN and Month.

As their name suggests, there is nothing in the Reduced set of files that cannot be found in the Pool set. The same data is to be found in both. Reduced is the result of applying a set of data reduction rules to Pool, that's all. The entire contents of Reduced can therefore be generated from Pool. This is a useful repair tool, in the event that Reduced has to be replaced, but more important is the fact that the data reduction rules could easily be varied. The only point of having Reduced files is to give speedy access to frequently required totals (with a small enough data set, the Reduced files would be unnecessary). With variable rules readily available as options within an accounting object, a developer could optimize access efficiency for each client system.

A batch is identified by the date and time posted. All the postings in the Pool carry this date/time stamp. I could have provided a generated BatchNum but, in another capricious binding decision, did not. The effect is the same – the Batch file contains the same data as Pool – you just use different data reduction rules to get from one to the other. This is another useful repair tool, and another set of data reduction rules that could be made variable.

Intermediate Processes

The structural mainstay of classic accounting is the Chart of Accounts. Firstly, it establishes a general ledger and (possibly) subsidiary ledgers. It classifies the accounts in each ledger into categories, such as capital (equity), assets, liabilities, income or expense. It describes the internal structure of each ledger, distinguishing normal accounts from control accounts. For each control account, it defines the normal accounts to be summed so as to agree with that control account's total. So, where is the Chart of Accounts in relation to the nucleus?

In my working application I have had to choose a Chart file with one particular layout and a hard-coded browse/form for its updating. That"s the way it is with procedural code when you have a living to earn. What *are* the variations that could invade a real-world Chart?

An important one is sub-accounts. And sub-sub-accounts. And so on. Some account codes are alphabetic, some numeric, some a mixture. Some are short and some are long. Account categories can also come in subtly different flavours. This variety of forms is a vital clue. It says the Chart *maps* different external realities. Better then, to think of a Chart as mediating between the outside world and the nucleus. This does not mean, however, that a Chart object is impossible, only that it would have to cope with greater variety than a nucleus object.

An interesting side-effect of having Functions represented right inside the nucleus (which makes the engine an implementation of matrix accounting) is that this simplifies the Chart of Accounts. I have been able to avoid sub-accounts altogether because of them, giving a simple flat structure to the Chart. Could this be a clue to devising a sufficiently simple Chart object? Perhaps this possibility can be explored later on.

Whenever a new record (for an Account) is added to the `Chart` file it is allocated a permanent `PAN`. Users never see this number and only know the account by the ledger, account code, category, name and type etc. that they assign to it. As a consequence, the code and name can be changed at any time. Similarly, when a new record (for a Function) is added to the `Funcs` file, it is allocated a permanent `FUN`. Users, however, see only the Function codes and names they enter and may change these at will.

When a batch of transactions is added to `Pool` and `Reduced` there are two steps involved. The first step converts the transactions (which have been entered using the codes known to users) into a queue of postings organised by `PAN`, `Stream`, `FUN` and `TSN`. The second step applies these postings to `Pool` and `Reduced`. In the real-life system, the function `PostTrans()` does the first step and `PoolAddMany()` the second.

`PoolAddMany()` can look at the `Chart` and `Func` files using the `PAN` and `FUN` keys. It particularly needs to refer to `Chart` as it is this file that tells the engine which normal accounts to add up to generate any required control account postings. For example, the accounts in a debtors' subsidiary ledger will all point to a debtors control account in the general ledger, and so on. By referring to `Chart` the `PoolAddMany()` function works out what additional control totals to add to `Reduced` for every batch it processes.

Extracting Data From The Pool

To start making this account more concrete, here is the code from a simple engine function, called `CurrentBal()`. As its name suggests, it returns the current balance of an account. The parameters are:

```
(PAN,Stream,FUN,LastTSN,AmtBal)
```

and the protoype is:

```
(ULONG,BYTE,ULONG,ULONG,*DECIMAL),LONG
```

The value of `LastTSN` is returned by the function `PoolTSN()` and is supplied to the `CurrentBal()` function as an input parameter. As you will see in the code below, `LastTSN` (if it is non-zero) limits the only LOOP in the function, implementing the fuzzy balance feature described above.

The balance returned (via `AmtBal`) may be the consolidated balance for all functions (`FUN` is zero) or the balance for one function (`FUN` is non-zero).

In the real-life application I call `CurrentBal()` directly when I want the balance of an account (for any stream, for any or all functions). In a more disciplined solution, there might be a wrapper function to mediate between developer and engine. A wrapper could hide `PAN`, `Stream` and `FUN` and allow the developer to think in the same codes the user employs.

There are around 40 engine functions in the application. Here is part of another one called `PoolBatch()`. Its purpose is to return (in a queue) all the postings that make up a batch. In the real-life application I use the contents of the queue to populate a temporary file so that the user can see it in a browse (how I wish Clarion had a template for directly browsing a queue!).

The parameters are:

```
(STRING,LONG,LONG,*SHORT,*DECIMAL),LONG
```

and the protoype is:

```
(Year,ThisDate,ThisTime,TranCount,TranProof)
```

`Year` may have the values "Current" or "Previous". Although there may be `Pool` files for as many prior years as you like, the implementation I have written limits read-write access to two years. `ThisDate,ThisTime` define one batch (by its date/time stamp). `TranCount,TranProof` are reply parameters.

`PoolQu` is defined as follows:

```
PoolQu QUEUE,PRE(QPL)
LIKE(PLA:Record) END
```

That is, it changes its fields automatically whenever I change the definition of a `Pool` file and re-compile. This allows me to fill a record in the queue with one instruction:

```
PoolQu = PLA:Record
```

Engine procedures directly reference very few `Pool` fields. In the above two examples, only key fields and the amount field are referred to directly. Where possible, the engine manipulates the `Pool` at the record level, using the method illustrated in the second example. This means that a developer can add fields to the Pool files without having to revise engine functions. The extra fields just go along for the ride, as it were. In the real-life functions, input parameters may refer to one or more named queues rather than relying on a global queue as shown above. Clarion’s ability to transfer whole queue-fulls of data gives the engine much better separation from the code that uses it.

As an example, consider the function `PoolGetStatementLines()`. It returns a queue of statement lines in the correct order, ready to include in a Debtor’s (or Creditor’s) open item statement. That is, the queue first lists invoices that have been paid, together with their matching payments and follows this with the unmatched invoices. The CPCS Report Writer allows printing from queues, so the returned queue can be dropped straight in. The prototype is:

```
(LineQu QLN, STRING, ULONG, *DECIMAL, *DECIMAL, *DECIMAL, pf↵
  *DECIMAL,*DECIMAL,STRING,STRING,LONG),LONG
```

The parameter list is:

```
(QLN,Ledg,WhichPAN,Days90,Days60,Days30,Current,Total,↵
AllOrCurrent,CurOrPrev,DateCut)
```

The input parameters are:

`LineQu`	is a named queue `(QLN).` The function fills it with the statement lines for one debtor or creditor account.

`Ledg`	is either "DR" or "CR", indicating a debtor or creditor. The distinction allows for some differences in processing
`WhichPAN`	gives the account, as it is known within the engine
`AllOrCurrent`	tells the function to return all the account's transactions, or just the current ones that would go into a normal statement
`CurOrPrev`	requests either the current year's transactions or the previous year's.
`DateCut`	supplies a cut-off date beyond which the function is not to go.

The reply parameters are:

`Days90, Days60, Days30, Current , Total`

These go straight into the aged balance line of a statement. Although named for a 30 day indebtedness cycle they are variable in content, and could be returning values for 21, 14 and 7 days of indebtedness, for instance. That there are four aged balances and not three or five is, of course, arbitrary.

Whatever the shortcomings, the function, which has been written at the generic, engine level, does most of the work involved in putting together a debtor or creditor statement. Other such functions do the hard work for a bank reconciliation, and so on.

Verifying And Manipulating The Pool

Having an engine to manage persistent data such in the `Pool`, `Reduced` and `Batch` files and giving it access to intermediate files such a `Chart` and `Funcs`, means that quite a few jobs can be done at this generic level.

For instance, verification need only be written once, as it can be fully expressed using the data structures understood by the engine. By verification I mean a process that checks a set of accounting records to see that they balance, that control accounts agree with subsidiary ledger totals and that, in effect, the records comply with correct accounting principles. As an aside, I should note the peace of mind it brings when a worried user rings up with a complaint, but is able to say "it passes the Verify Run". As soon as these words are out, I know the problem is in his use of the system and not in my software.

As mentioned earlier, the engine can replace the `Batch` and `Reduced` files using only the data in the `Pool`. So, there are functions such as `RefreshBatch()`and `RebuildBalances()` to do this work. There is also `CheckCHART()` to test the internal consistency of the Chart of Accounts.

End of period logic can also be done by the engine. The function `PoolClose()` ends a financial year and `OpenNewYear()` does exactly what it says. `PoolEndMonth()` puts a month's set of bank reconciliations to bed, while `PoolEndDrCrPeriod()` removes already matched invoices and payments.

To make a more general statement, any accounting work that can be coded in a form understandable to the data structures within the engine can be done there and does not have to be written elsewhere. One of the outcomes of converting my procedural code to objects will be to extend the scope of the engine, and therefore increase its functionality.

Summary

Perhaps you did not expect an ending quite so soon? Where are these precious objects, you may be wondering. Well, my aim has been to put a question and hope that some of you may respond with thoughts and suggestions of your own.

I believe that accounting has a stable nucleus, that it is not a chaotic mess. Further, that its central and enduring principles point the way to the design of useful accounting objects. These objects could carry out accounting in a completely regular manner – following their own rarified, abstract representations of the accounting process. Intermediate objects could mediate between them and the peripheral processes visible to the user.

The leverage of each hand-written line of code has to keep on increasing. Wizatrons do this at the busy, noisy periphery. Templates do it somewhere in the middle. Could accounting objects do it for the most central parts of an accounting system? I said in the opening sentence of this article that I have never written or heard of an "accounting object". This remains true, but not for much longer, I hope.

Join The Open Source Accounting Project

If accounting objects interest you, I hope you'll join us in the **clarionmag.opensource.accounting** newsgroup on the ClarionMag news server (remember to refresh your newsgroup list) as we work toward developing useful, flexible accounting objects. All accounting code to come out of this project will be covered under the Developers Open Source Public License. If you have questions about getting access to the news server please refer to your subscription confirmation email or send an email to webmaster@clarionmag.com.

Dave Harms, Editor

David Podger has been an independent Clarion developer in Australia for a decade. He presently lives in Katherine in the Northern Territory, where he sells a specialised accounting application for remote communities.

Main Menu | Log In | Subscribe | Links | FAQ | Advertising

Clarion MAGAZINE

Clarion
Development
Resources

published by
CoveComm Inc.

Main Page

Log In
Subscribe
Renewals

Frequently Asked
Questions

Site Index
Article Index
Author Index
Links To
Other Sites

Downloads
Open Source
Project
Issues in
PDF Format
Free Software

Advertising

Contact Us

Search

News

Clarion News

February 29, 2000

BLOB Compression/Decompression Demo

Linder Software has a new set of BLOB compression and decompression source code demos available for download. Demos include adding and restoring images (standard and in-memory) and adding and restoring text (in-memory).

Updated Catalyst Prototypes

Leonid Chudakov’s updated prototypes for Catalyst Socket Tools version 3.2 are now available.

Gitano Software FAQ

Gitano Software has a new FAQ site which will be updated as support questions arrive.

February 22, 2000

Windows-Like Menus For C5.5 Web Edition

Emerald Technology is a template add-on that allows a Clarion 5.5 web app to have drop-down menus like a normal Clarion application. Available at www.clarionshop.com.

TX Control Class Documentation

The TTX control class wrapper now comes with documentation.

Source For Gitano Software Products

Gitano Software has announced that effective immediately, only sales through their web page will be honored. There are no authorized distributors for Gitano Software products.

Clarion Freeware From Sterling Data

Two new freeware items have just been added to the Clarion Software Library. Demo Maker is a template which converts any application into a time-limited demo. by Alex Oberhard. Also by Alex, a DOS (CPD) LEM for fast graphics drawing, complete with example app.

February 15, 2000

Last Chance To Renew And Save!
(Feb 29,2000)

Last Chance To Save On A New Subscription!
(Feb 29,2000)

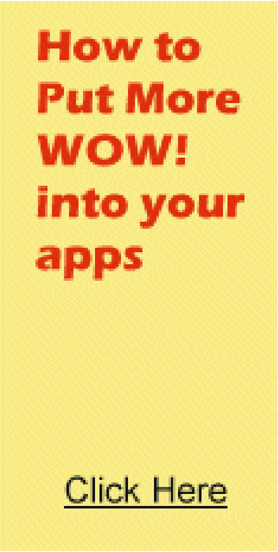
February 2000 News
(Feb 29,2000)

The Novice's Corner: Clarion Code 4
(Feb 29,2000)

List Box Marking
(Feb 29,2000)

Open Source Update: Thread Manager
(Feb 29,2000)

Read The January 2000 News



DeveloperPlus
Your source for development tools and add-ons.
Your outlet for application sales.

G-Cal 2.0 Released

Gitano Software has released G-Cal 2.0; more calendars, more options, increased functionality, and a new registration wizard. If you have purchased G-Cal in the last 30 days, the upgrade is free. All other registered users of G-Cal can upgrade for a nominal fee. These calendars and date/time functions will not be included with GCal Pro. GCal and GCal Pro will remain two separate packages.

PD Date Tools Update

ProDomus expects a February beta release of its new class-based PD Date Tools. New features include a calendar control and drop calendar, and a one-step global extension that populates an extension on every window.

Updated Country Format Information

Sterling Data's country format information page now includes web page collections courtesy of John Barron. These include various country information pages, address formats for Universal Postal Union members, and Microsoft's recommendations for 30 countries.

Debugging Freeware

Wayne Haynes has created a freeware debugging tool called The Programmer's Billboard which uses DDE to log test messages without disrupting the flow of the program under test.

Open Linux For TopSpeed Update

Ron Schofield is back on line with the Open Linux project. Anyone interested is asked to update their information on the members page and fill out any polls on the web site.

February 8, 2000

ClearICE for BlackICE Defender

ClearICE is a reporting utility for users of BlackICE Defender. Written in Clarion by Ben Brady, this program extracts attacker information from the BlackICE logs.

Clarion Handy Tools Build N

New features in Clarion Handy Tools Build N include a source code version of all the tools, 110+ pages of help material in HLP and HTML format, support for C5.5, new and revised example apps. The web site has also been updated web site.

Wild Wild Wares Template Site

This site has a large variety of templates including autosize column, edit-in-place, browse coloring, control the number of app instances, math functions, ASCII import, child record removal, easy reporter, multimedia/display features, distance between two zip codes, system functions, string operations, security, and more.

Clarion 5.0 Review

This WinPlanet review calls Clarion 5.0 a "real jewel" and "pound for pound the best in its class."

Subscribe To TopSpeed's Newsletters

TopSpeed emails quarterly and biweekly newsletters – to get yours just visit this page and provide your email address.

Bruce Johnson To Teach Course At etc2000

Bruce Johnson will be presenting a day-long training course on the last day of the East

Tennessee Clarion Conference and Gathering. The course will cover the practical side of embedding in ABC templates.

NetTalk 1.0 Beta 4 Released

NetTalk, a set of tools that allow your programs to easily communicate with each other over TCP/IP networks, is now in Beta 4. Includes prebuilt objects for file transfer, remote app closing, workstation time synchronization, and chat. New in Beta 4: WAN support, multi-dll support, and bug fixes and template improvements. NetTalk will usually cost $299, but is $199 for the duration of the beta program.

Clarion 5.5 Beta 2 Coming Soon

Clarion 5.5 B2 is nearly ready – word is the beta should be out within a couple of weeks, and will be available for download and by CD.

Send Your Mug For A Chance To Win

Mike McLoughlin of Sterling Data is looking for more developer pictures for his rogue’s gallery. Send a GIF or JPG of yourself with a caption by Friday, Feb 11/2000 and you’ll be eligible to win a free copy of Mike’s contact manager Marketeer USA (written in Clarion of course)

The Clarion Insider

Paul Attryde’s Clarion OnLine articles, as well as other articles, tips, and shareware are available at this web site. Also includes links to other sites of interest to Clarion developers.

Send Internet Mail 3.06 Released

This release of the Send Internet Mail template includes sender address bug fixes.

Clarion MAGAZINE
Clarion Development Resources

published by CoveComm Inc.

Feature Article

February, 2000

Last Chance To Renew And Save!
(Feb 29,2000)

Last Chance To Save On A New Subscription!
(Feb 29,2000)

February 2000 News
(Feb 29,2000)

The Novice's Corner: Clarion Code 4
(Feb 29,2000)

List Box Marking
(Feb 29,2000)

Open Source Update: Thread Manager
(Feb 29,2000)

Understanding Clarion Code Part 4

by David Harms

In the previous article in this series I modified Carl Barnes' utility for deleting temporary files by adding a window to display the files to be deleted. Now it's time to take things a step forward and look at how list boxes actually format data.

As well, I've switched the project from 16 to 32 bits and am using a directory queue compatible with long file names. More on that later.

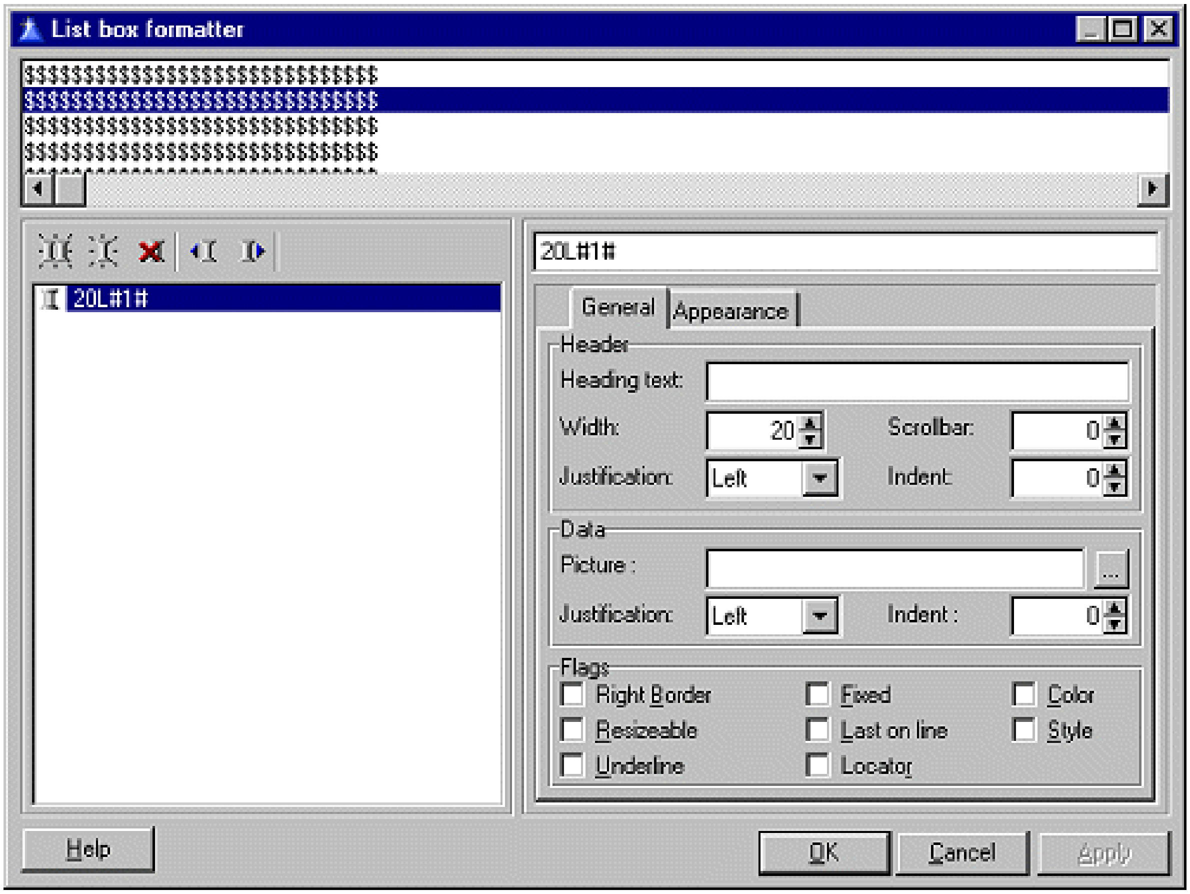
So far the list box for the utility is declared in the window structure without any formatting information:

```
LIST,AT(12,12,216,116),USE(?List1),↵
  HVSCROLL,FROM(DirQ)
```

The list box will still display the data from the queue, but it's going to look ugly. Without any format specifier the list box just treats the queue's fields as string fields and throws the raw data up on the screen, using arbitrary field lengths. It isn't pretty, but it's workable and a useful thing to keep in mind if you ever need just the barest of list boxes.

The easiest way to prettify the list box is to use the list box formatter. Invoke the window formatter (press Ctrl-F with the cursor inside the window source), right click on the list box, and choose List Box Formatter from the context menu. You should see something like Figure 1.

Figure 1. The list box formatter.



List box formatter

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

20L#1#

20L#1#

General | Appearance

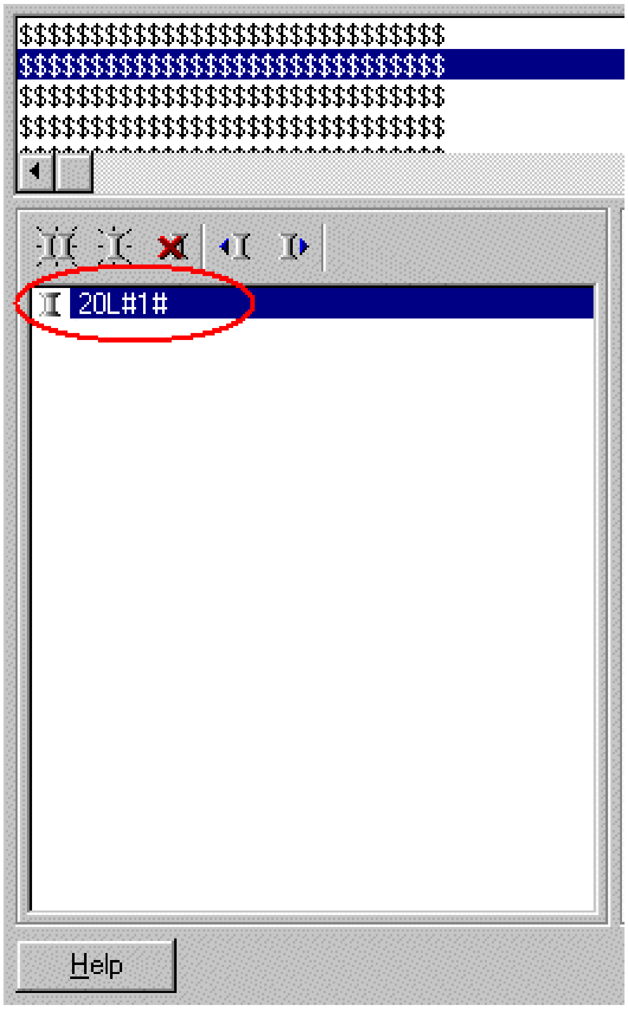
Header
- Heading text:
- Width: 20
- Scrollbar: 0
- Justification: Left
- Indent: 0

Data
- Picture :
- Justification: Left
- Indent : 0

Flags
- □ Right Border
- □ Resizeable
- □ Underline
- □ Fixed
- □ Last on line
- □ Locator
- □ Color
- □ Style

Notice that the formatter has automatically dropped in a first field for you, and that's as good a place to start as any.

Figure 2. The default first field in the list box formatter.



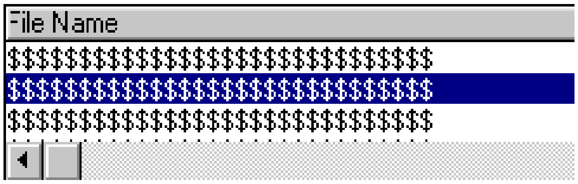
You’ll want your list box to correspond to the queue which stores the directory information.

Figure 3. The directory queue.

```
FILE:Queue   QUEUE,PRE(FILE),TYPE
Name           STRING(FILE:MaxFileName)
ShortName      STRING(13)
Date           LONG
Time           LONG
Size           LONG
Attrib         BYTE
             END
```

Start with the `name` field. In the list box formatter, enter File Name in the Heading Text field. As you do notice that a column header appears with the heading text. If you look closely at the header you’ll see that the part of the "F" is cut off.

Figure 4. Header formatting problems.



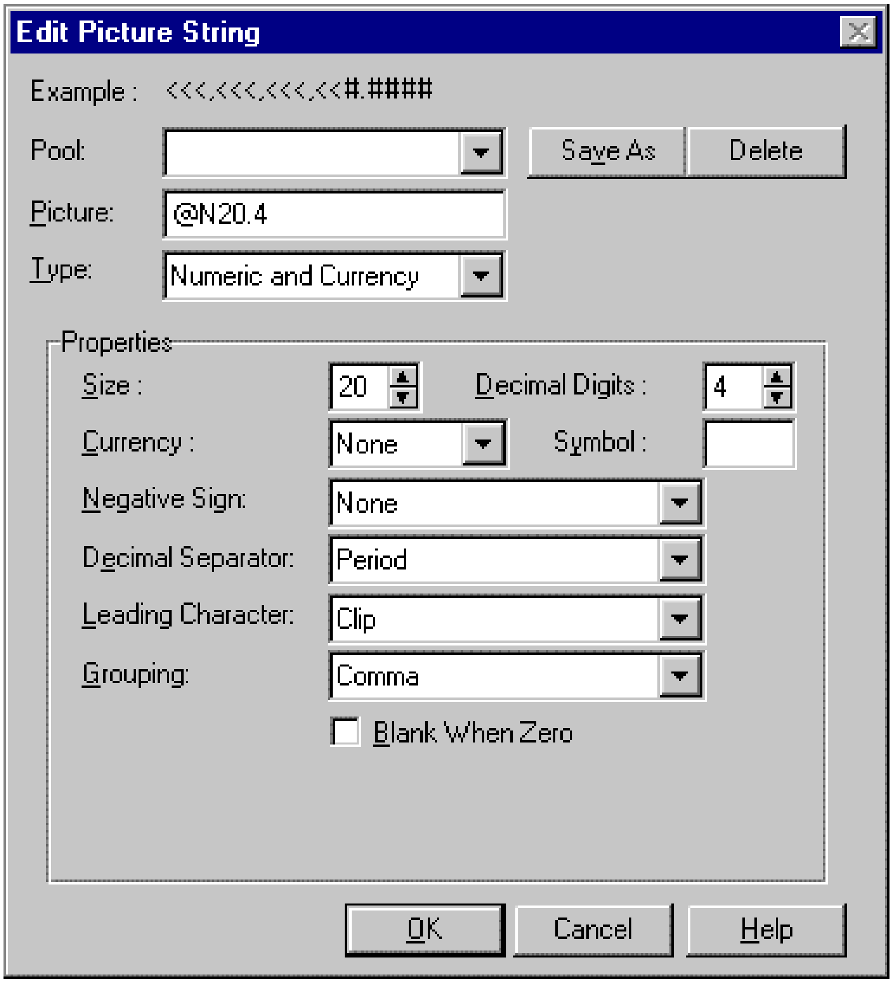
There are several ways to deal with this problem. The General tab has two groups of settings, one for the header and another for the data. In the header group you can another alignment, such as Center. Or you can set the header indent to a value such as 2, which shifts the text far enough right to avoid the overlap with the edge of the header. Do the same for the data field.

You should also specify a picture in the Data group. Since this is a large string field, `@s255` will work nicely. You can adjust the actual width of the fields by dragging the column separators once you have all the fields added, but as a rough guideline the width value (which is in DLUs – see Dialog Units in the Help) should be about four times the number of characters, for a string field. In this case the width will be considerably less than the picture's maximum.

Fancy Formatting

If you're not sure of the picture you want to use for a given field, click on the ellipsis button next to the Picture field to invoke the picture editor.

Figure 5. The picture editor.



Example : <<<,<<<,<<<,<<#.####

Pool:

Picture: @N20.4

Type: Numeric and Currency

Size : 20

Decimal Digits : 4

Currency : None

Symbol :

Negative Sign: None

Decimal Separator: Period

Leading Character: Clip

Grouping: Comma

□ Blank When Zero

The picture editor guides you through the many options available for formatting data. Figure 5 shows the options for numeric and currency types.

> **TIP:** If you find yourself creating the same pictures over and over, click on the Save As button. This will write the picture you've chosen to the c5pict.ini file in your Clarion5 bin\ directory. You can then choose it from

the Pool drop list.

Back at the list box formatter, have a look at the flags group. I usually check Right Border and Resizable. In most cases I find underlining makes the list box too busy. I'll get to the rest of the options later.

One nice feature of the formatter is that it remembers the last flags set. So go ahead and add a second field (press Insert or click on the button.

The second field in the directory queue is a `STRING(13)` for the short file name. You can set this up similarly to the first field, though of course you won't need a picture larger than `@s13` and the display width can be much less.

The third field in the directory queue is a `LONG` which carries a Clarion date value, so you'll want an appropriate date picture such as `@d2`. Similarly choose a time picture for the fourth field, and a numeric picture for the fifth field, one large enough to accommodate whatever maximum file sizes you're likely to see.

You can add the `Attrib` field if you want, but that's actually a bitmapped field, so unless you're good at decoding bit-mapping by sight you might as well leave it off for now. Although the list box will display all fields if you don't specify a format, once you specify fields it won't show any not referenced by the format.

Saving/Restoring The Format String

Compare the new `LIST` statement with the original. The difference is the new `FORMAT` attribute:

```
FORMAT('199L|M~File Name~L(2)@s255@52L|M~Short Name~L(2)' |
  & '@s13@49L|M~Date~L(2)@d2@35L|M~Time~' |
  & 'L(2)@t4@20D|M~Size~L(2)@n12@')
```

Format strings can look pretty intimidating, which is why there's a list box formatter to make creating and altering them a lot easier. But because they are just strings, you can read and write them on the fly using `PROP:Format`.

When I formatted the list I checked the Resizable box for each column so the user would be able to adjust the column widths to taste. But what's the point of letting your users do that if you keep resetting the widths when the program starts up? Wouldn't it be better to save and restore those settings?

As it turns out, this is very easy to do with the `GetINI` and `PutINI` methods, which read/write the INI file you specify. Add the following code after the `open(Window)` statement:

```
?List{prop:format} = GetIni('Test','ListFormat',|
?list{prop:format},'.\deltemp.ini')
```

The third parameter of the `GetINI` function is for a default value. In the event that this is the first time this program is being run on this machine, I want to be sure that the values specified in the list box's `FORMAT` attribute are used. You can use `PROP:Format` to read and write the list box format.

Now add this code before the `CLOSE(Window)` statement:

```
PutIni('Test','ListFormat',?list{prop:format},|
  '.\deltemp.ini')
```

This will save the current settings to the INI file. I prefix the name of the file with `.\` to force the file to be created in the current directory, rather than in the Windows directory.

Now you can run the program, adjust column widths, exit, relaunch and the list box will have the same appearance as when you exited.

Since the format string determines almost everything about the list box's appearance, you can also use this approach to define a variety of standard formats and let the user decide which look they want. A good example of this is Carl Barnes' Clarion Source Search program which offers a number of predefined list formats for the search results window. You could combine these techniques to let users pick a basic look and then save any changes they make.

Field Ordering

When you're working with browses in the Application Generator you can move fields around in the browse as much as you like. When you're hand coding, moving fields around may have unexpected effects. If you make the Short Name field the first field in the list box and the (long) File Name field the second field and run the app, the order of the data remains unaffected. This is because the list box formatter, when used on source code, doesn't preserve any field ordering information. In the example app the format string for the Date field appears in the formatter as

```
70L(2)|M~Date~@d2@#3#
```

The `#3#` at the end of the string tells the list box that this field corresponds to the third field in the associated queue. If you change the order of display, or only wish to display certain fields, you need to update these values so that the correct data is retrieved from the queue. The problem is that these values are not preserved by the formatter when working *outside the AppGen*. If you're in the AppGen working with a browse you won't see this problem.

If you need to change field orders or skip queue fields in hand-coded lists you'll have to modify the format string directly, or use the property syntax.

Using The Property Syntax

If you don't want to change an entire format string using `PROP:Format`, you can change individual column settings using various `PROPLIST` properties. For instance, to specify which queue field a column should get its data from, use `PROPLIST:FieldNo`. The code to swap the first two fields looks like this:

```
?list{proplist:FieldNo,1} = 2
?list{proplist:FieldNo,2} = 1
```

Groups And Multiple Lines

You can group fields under a common heading, if you wish. In the list box formatter highlight the first field you want in a group and click on the New Group button (or press Shift+Insert). You can then move additional fields into the group using the up/down arrow buttons.

If you'd like to display a queue record across multiple lines, you first have to create a group. Then choose the field in the group before the one you want on the next line, and check Last On Line. You can also set this value for a column at runtime using

`PROPLIST:LastOnLine`.

Wait, There's More!

I've barely scratched the surface of what you can do with list boxes and format strings. Have a look in the help for FORMAT (set LIST or COMBO layout) for a more complete description. I'll just touch briefly on a few areas you may want to pursue further.

You've probably seen Clarion list boxes that use colors. The Application Generator has fairly good support for colors in list boxes, and will automatically add the four LONG fields that must be added to the queue for each field to be colored. These fields contain values for foreground and background colors in normal and selected modes. As with other list box formatting, you can adjust these values using the property syntax as well.

To get a bit more control over your list box's appearance consider using styles. Essentially you set up an array of styles, and then tell the list box which style to use for a particular field (or column). This is a bit more efficient than colors when you have just a few colors in use (as is most likely the case) and has the added benefit of letting you change font styles as well. Whereas colors require four additional LONG fields for each colored field, styles only require one additional LONG per field. For more on styles see Mike Pickus' Advisor article.

The Ultimate In List Box Control

If all of that isn't enough for you, have a look at virtual list boxes. There isn't a lot of documentation available on this subject, so you're mostly on your own. But there's an example in the Help (see PROP:VLBProc under Runtime Properties) to get you started. Essentially a virtual list box lets you insert your own procedure (or class method) to handle the formatting of the list data. This procedure is called by the runtime library for each cell to be displayed. If you're using styles or colors, it will also be called for each color or style field. Virtual list boxes are one of the most powerful and least-used aspects of the Clarion language.

Summary

One of the things that endears the Clarion language to developers is the ease with which features can be implemented. Even in hand code, list boxes, which can appear very complex, are relatively simple to manage. You can use the property syntax to set the format of the entire list box or individual columns. Styles make it easier to control list formatting. And if you want to really get your hands dirty you can always use the virtual list box.

Download the source

Understanding Clarion Code Part 1

Understanding Clarion Code Part 2

Understanding Clarion Code Part 3

Understanding Clarion Code Part 4

David Harms is an independent software developer and the co-author with Ross Santos of Developing Clarion for Windows Applications, *published by SAMS (1995). He is also the editor and publisher of Clarion Magazine.*

Main Menu | Log In | Subscribe | Links | FAQ | Advertising

Clarion MAGAZINE

Clarion
Development
Resources

published by
CoveComm Inc.

Main Page

Log In
Subscribe
Renewals

Frequently Asked Questions

Site Index
Article Index
Author Index
Links To Other Sites

Downloads
Open Source Project
Issues in PDF Format
Free Software

Advertising

Contact Us

Search

Feature Article

February, 2000

List Box Marking

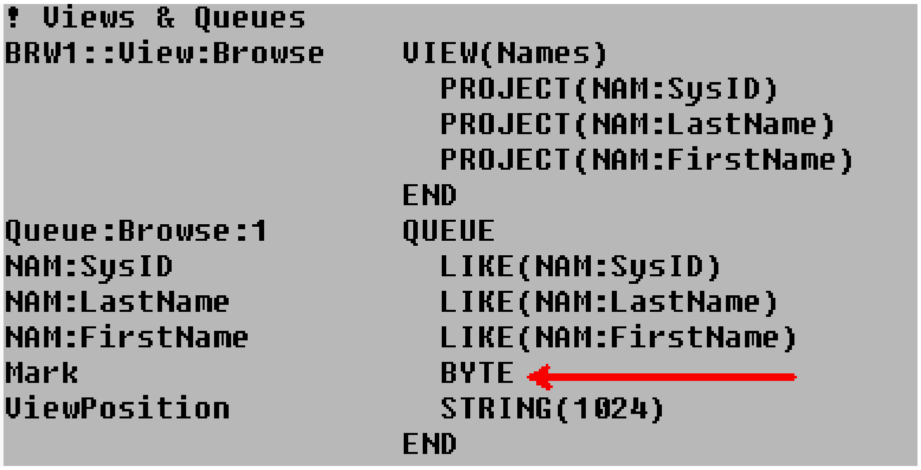
by Steve Parker

Windows provides the ability to select and highlight records in a list. This feature is called "marking." Clarion builds in support for this by providing a column in the queue which is used to form the browse. This field is called, originally, "Mark" and may be addressed as Queue:Browse:1.Mark (see Figure 1).

Searching for "mark" in the on-line help gives instructions for implementing Windows marking:

> The MARK attribute (PROP:MARK, write-only) enables multiple item selections from a LIST or COMBO control. When an item in the LIST is selected, the appropriate flag field is set to true (1). Each marked entry is automatically highlighted in the LIST or COMBO. Changing the value of the flag field also changes the screen display for the related LIST or COMBO entry.
>
> If the MARK attribute is specified on the LIST or COMBO, the IMM attribute may not be.

Figure 1. The Mark field in the queue.



```
! Views & Queues
BRW1::View:Browse     VIEW(Names)
                        PROJECT(NAM:SysID)
                        PROJECT(NAM:LastName)
                        PROJECT(NAM:FirstName)
                      END
Queue:Browse:1        QUEUE
NAM:SysID               LIKE(NAM:SysID)
NAM:LastName            LIKE(NAM:LastName)
NAM:FirstName           LIKE(NAM:FirstName)
Mark                    BYTE
ViewPosition            STRING(1024)
                      END
```

The key to implementing marking is that the mark attribute must be placed on a list, naming the queue field holding the mark as a parameter:

Last Chance To Renew And Save!
(Feb 29,2000)

Last Chance To Save On A New Subscription!
(Feb 29,2000)

February 2000 News
(Feb 29,2000)

The Novice's Corner: Clarion Code 4
(Feb 29,2000)

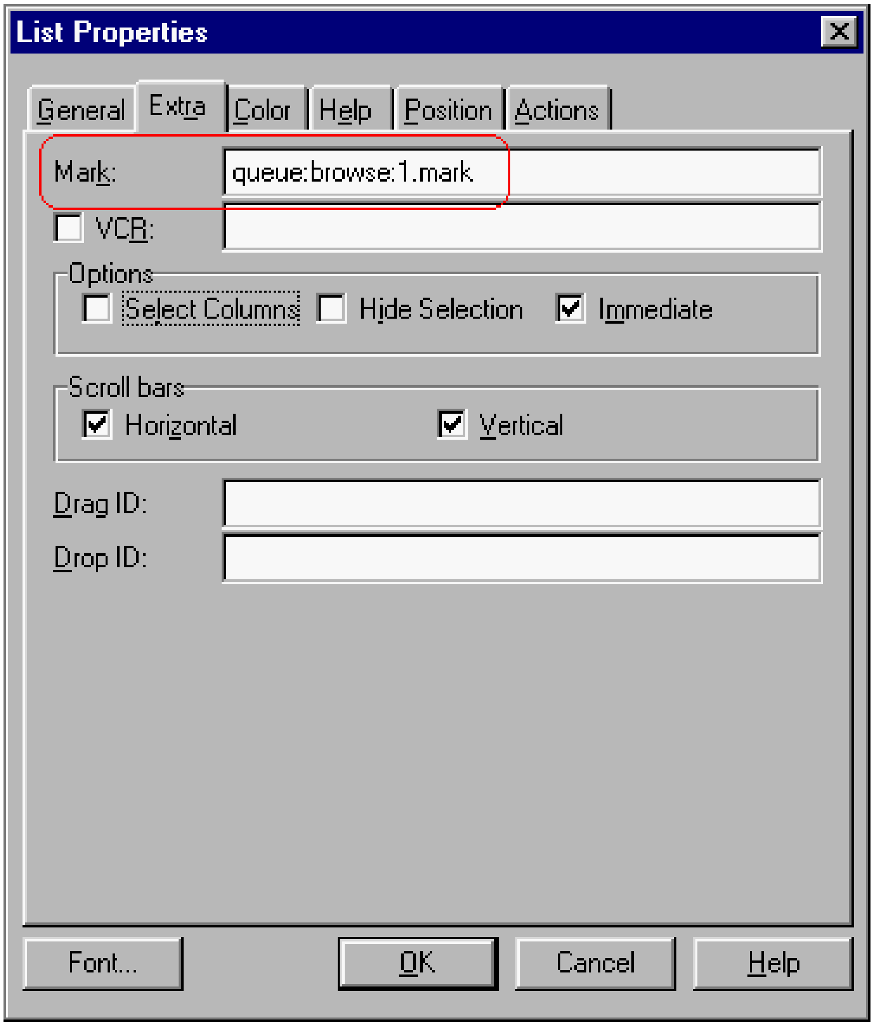
List Box Marking
(Feb 29,2000)

Open Source Update: Thread Manager
(Feb 29,2000)

```
LIST,AT(8,20,216,124),USE(?Browse:1),HVSCROLL,|
 MSG('Browsing Records'),FORMAT('45R(2)|M*@n-14@80L(2)|M' |
'~Last Name~@s20@80L(2)|M~First Name~@s20@'), |
MARK(Queue:Browse:1.Mark),FROM(Queue:Browse:1)|
,#SEQ(1),#ORIG(?List), |#FIELDS(NAM:SysID,NAM:LastName,NAM:FirstName)
```

There is a prompt IDE or this code may be added manually via the window's ellipsis.

Figure 2. Setting the Mark attribute for the list box.



Mark: queue:browse:1.mark

☐ VCR:

Options
☐ Select Columns ☐ Hide Selection ☑ Immediate

Scroll bars
☑ Horizontal ☑ Vertical

Drag ID:

Drop ID:

So there are three steps to implement marking:

1. from the Source button, find the name of the queue
2. add the Mark attribute to the list declaration
3. remove the IMM attribute from the list box

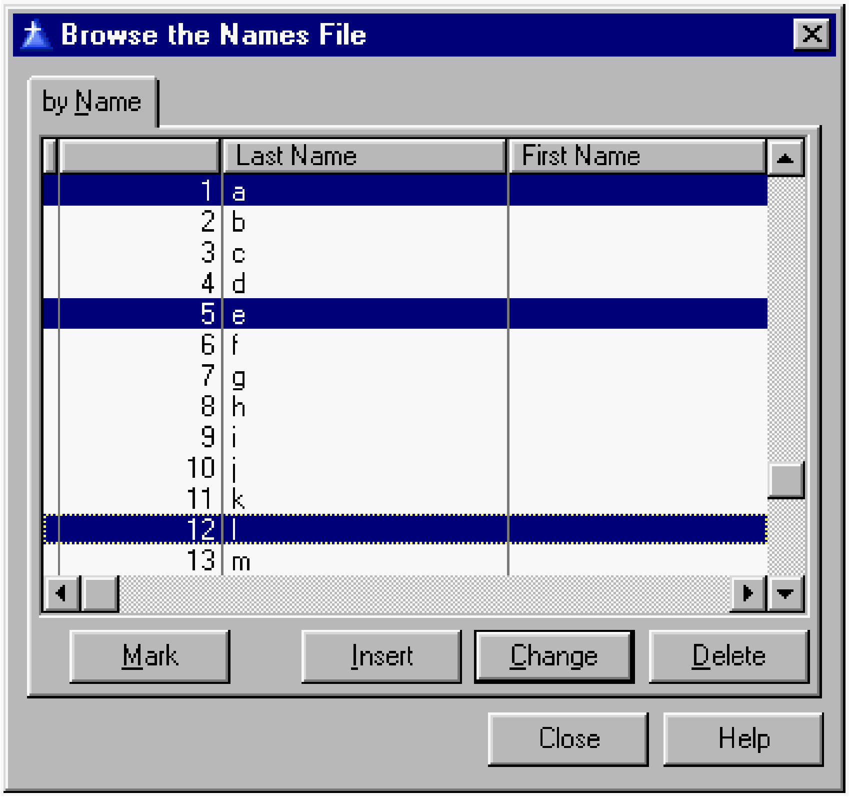
Simple.

Except that functionally, it is worthless. Or so it seems to me.

What You Get

The specification says that a selected record will be marked. And, indeed, if I select a record, it will be marked:

Figure 3. Default marking behavior.

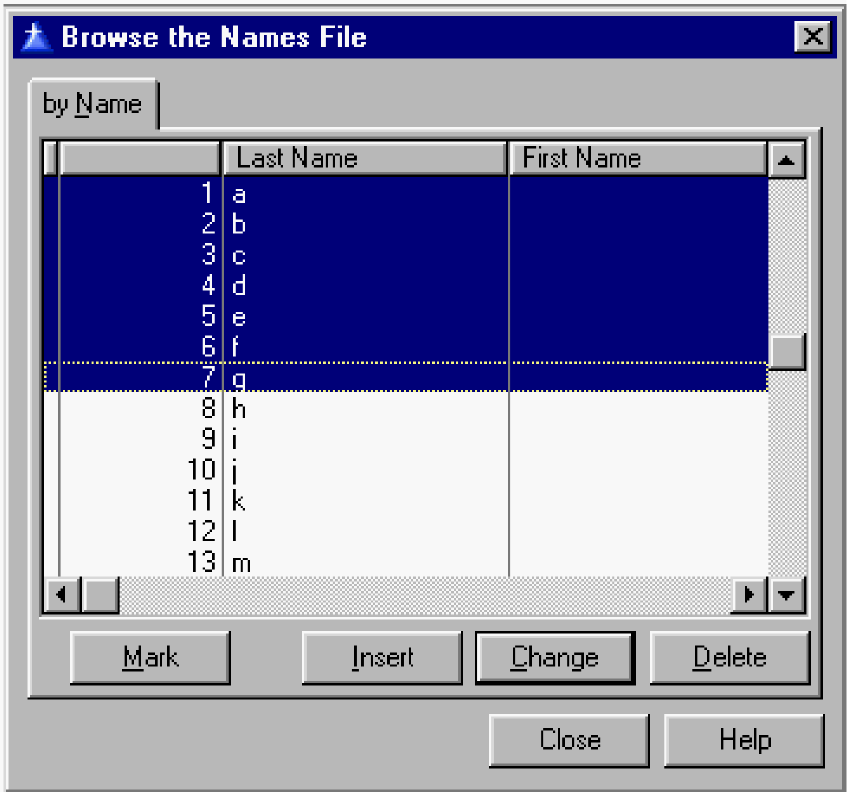


	Last Name	First Name
1	a	
2	b	
3	c	
4	d	
5	e	
6	f	
7	g	
8	h	
9	i	
10	j	
11	k	
12	l	
13	m	

Notice that the first record, having been selected when the browse is opened, is automatically marked.

I can mark by simply pressing the down arrow:

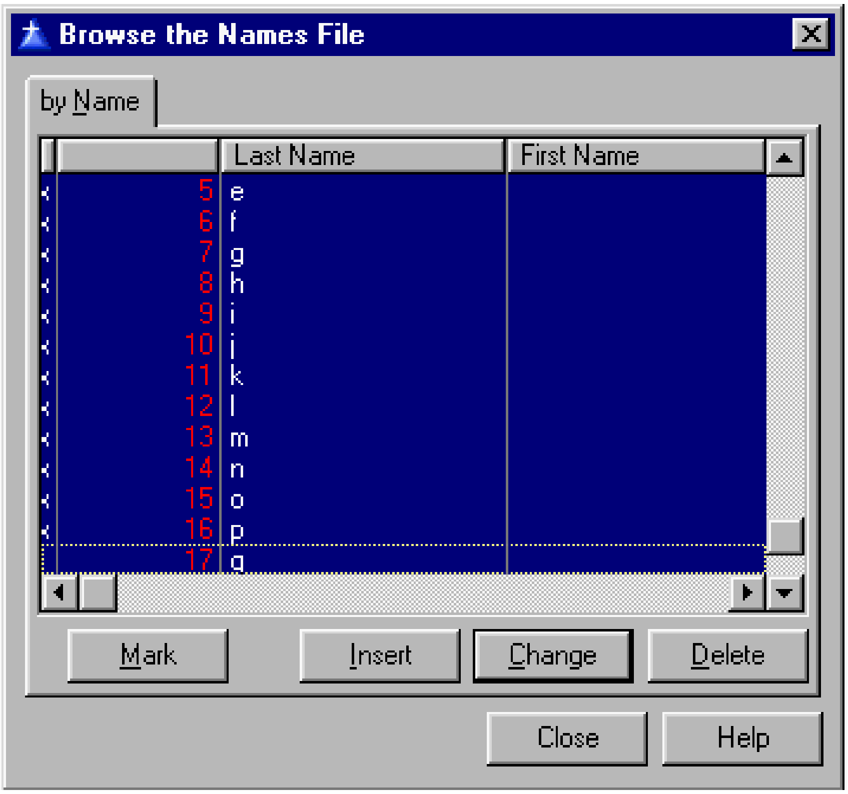
Figure 4. Marking a range of records.



	Last Name	First Name
1	a	
2	b	
3	c	
4	d	
5	e	
6	f	
7	g	
8	h	
9	i	
10	j	
11	k	
12	l	
13	m	

or by adding a record:

Figure 5. Adding a marked record.



	Last Name	First Name
5	e	
6	f	
7	g	
8	h	
9	i	
10	j	
11	k	
12	l	
13	m	
14	n	
15	o	
16	p	
17	q	

All records touch by the cursor are marked.

Pretty nasty. And it gets worse.

If I scroll down, calling a new page, marks are re-initialized. That is, existing marks are lost.

This happens because the Clarion runtime library, which manages the list box, refills the queue for the next page. (Remember, the queue only has enough entries to fill the list box; therefore, those entries, or rows, are re-used.)

As I said, it’s worthless.

Let’s Make A Spec

It is entirely possible to go mucking about in the code, trying to fix this. With sufficient trial and error, something functional might (possibly) fall out the other end. Or, I can think about it and explicitly describe how I want marking to function. Then I can just do it myself, be done in a few hours and know who is responsible when it doesn’t work.

I want to be able to select one or more records. And, I do not want to do it by simply touching the record in the list box (which is what the built in mark does). I want to mark by an explicit action, such as a button press or key combination.

I want to be able see the records marked.

I want the marks to be useable outside of the browse (obviously, Windows’ marks, which depend on the browse queue, cannot).

Oh yes, I want the marks to survive scrolling.

To recap:

- select a record under keyboard/mouse (program) control
- show the selections
- make the selections persistent.

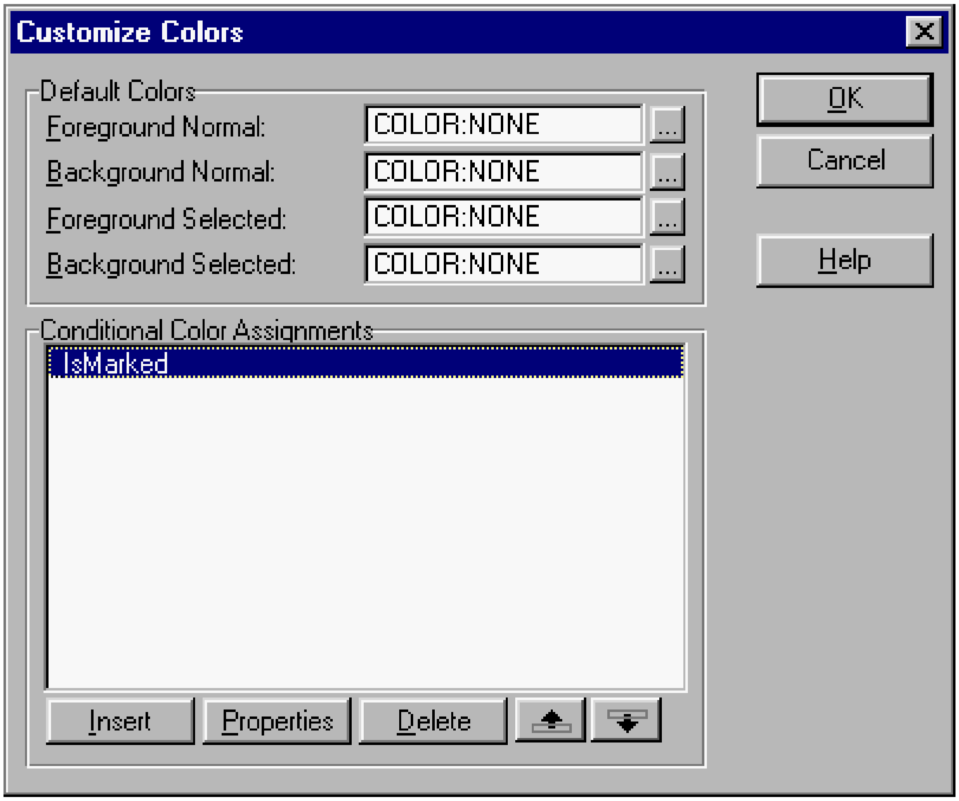
In fact, this is quite easy.

Anyone who’s used Clarion for any length of time can populate a button and use it to call a procedure. More advanced (slightly more advanced) users can set the ALRT() attribute and call a procedure in the Alert Key embed. Obvious candidates, in the current case, include Ctrl-MouseLeft, Shift-MouseLeft and a simple MouseRight. So the selection problem is solved.

Highlighting the selected record(s) is somewhat more difficult, as there are three possible methods for doing this.

1. I can place a local variable in the list and, if the record is marked, display an "X" or other symbol.
2. Likewise, I can turn on icons for the browse box and display an icon.
3. I can use conditional coloration of the list box to emulate what Windows does.

Figure 6. Setting conditional coloration.



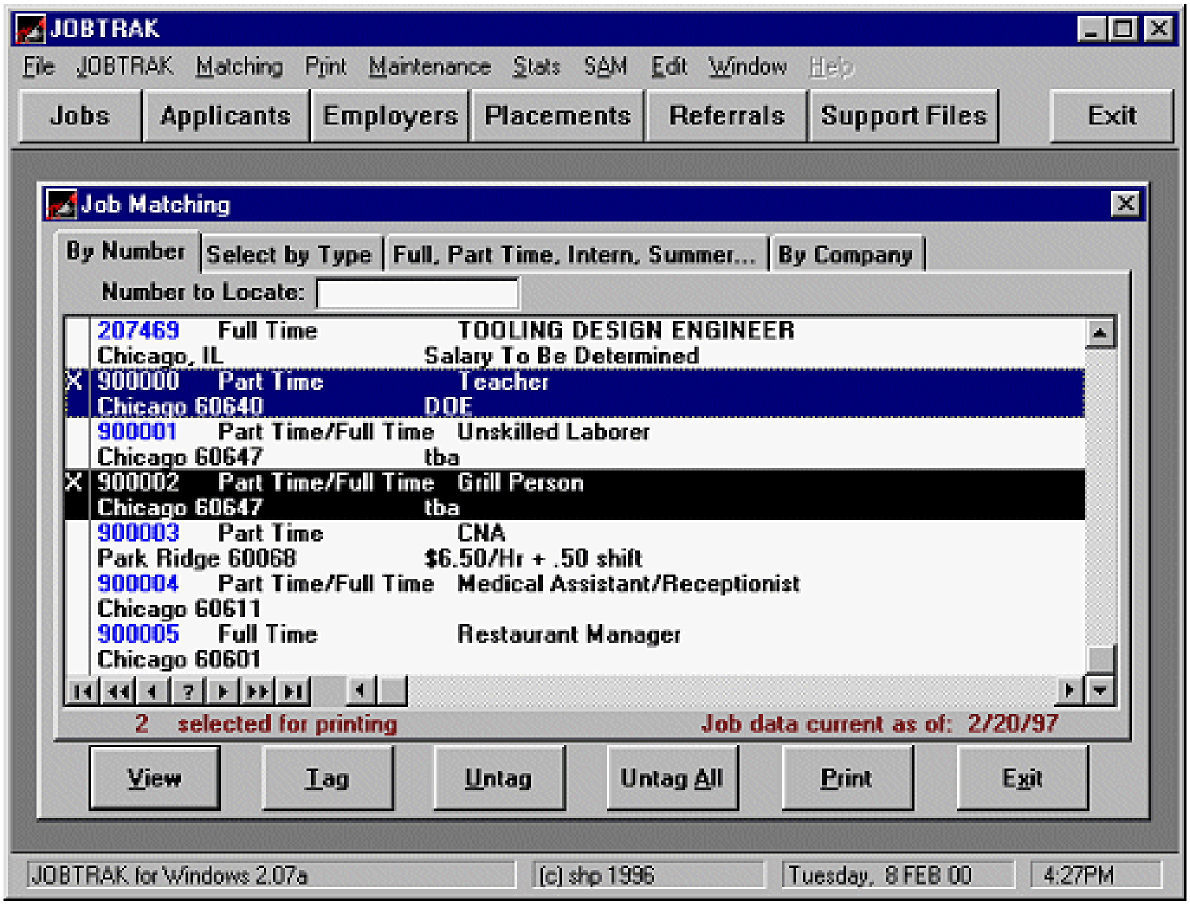
Foreground Normal: COLOR:NONE
Background Normal: COLOR:NONE
Foreground Selected: COLOR:NONE
Background Selected: COLOR:NONE

IsMarked

At first glance, the last option is the most appealing. After all, it demonstrates standard Windows behaviour. On the other hand, not only does it look just like what Windows does, but: (1) depending on the color scheme the end user has chosen, s/he may not be able to distinguish a marked record from the record currently selected in the list and (2) conditionally coloring each column in a list is more work than either of the alternatives.

It’s your call.

Figure 7. An application with conditional coloring.



Number to Locate:

```
  207469   Full Time            TOOLING DESIGN ENGINEER
  Chicago, IL              Salary To Be Determined
X 900000   Part Time            Teacher
  Chicago 60640            DOE
  900001   Part Time/Full Time  Unskilled Laborer
  Chicago 60647            tba
X 900002   Part Time/Full Time  Grill Person
  Chicago 60647            tba
  900003   Part Time            CNA
  Park Ridge 60068         $6.50/Hr + .50 shift
  900004   Part Time/Full Time  Medical Assistant/Receptionist
  Chicago 60611
  900005   Full Time            Restaurant Manager
  Chicago 60601
```

2 selected for printing

Job data current as of: 2/20/97

JOBTRAK for Windows 2.07a | (c) shp 1996 | Tuesday, 8 FEB 00 | 4:27PM

All that remains is making the marks persistent.

"Persistent." Does that mean from page to page, from procedure to procedure or from one instance of an app to the next?

Suddenly, this isn’t sounding quite so easy any more.

Tagging

The truth is that if marks can be made to persist as the end user scrolls the file from beginning to end, any other definition of "persistence" can be handled in a straightforward manner.

The important thing is to realize that the built in marking ability is poorly designed. Not only is it part of the queue record but there is no access to where and how it is set. Both of these are significant issues.

Suppose, instead, that I had a real variable indicating whether or not the record was marked. That would address all of the issues I have raised here.

I have control over a real variable: I can set it; I can interrogate it. And I can interrogate it in other procedures, like reports. I can clear it; I can ignore it. I can tromp it; I can stomp it.

Byte Me

The simplest (and, ultimately, least advisable) thing to do is add a byte field to the record structure.

Suppose I have a field, call it `CUS:IsTagged`, I can set it and I can clear it. This is simple programmer controlled I/O.

If `CUS:IsTagged` has a value, the record is tagged and, if not, it isn’t. (See? I told you this was simple.)

Because I don't want to display a 1 or 0 (or blank), I can use CUS:IsTagged as the condition for browse coloring or for displaying an icon. (See? I *told* you this was simple.)

This technique is not advised for the same reason it is so simple: the tags are stored in the data file. This means that any user who tags a record, tags it for every user. Therefore, if the app could be used by multiple users, this will probably not do. If only one person will use it (and I do use this technique in my batch compiler, for example), this is the easy way to mark records (actually, having used Mike Hanson's tagging templates since ... well, since there were Mike Hanson's tagging templates, letting someone else do it for you *is* easier).

However, this points me toward a more general solution.

When interrogating a byte field in the record structure, what is it I am actually using? In fact, I am not using the record; all I need is the byte and knowledge of, a reference to, the record. And, happily, Clarion gives me three ways to refer to a record: Pointer(), Position()and by a unique key value.

Position() *can* return incorrect records. In, for example, a multi-user environment, I capture the Position() and another user deletes the record. In this case, I may get the next record. This, I do *not* want.No record or an error is fine, but not the wrong record.

So Position() is not suitable for my purpose.

Pointer() is supported for all PC file systems and, obviously, a unique key value will also work. However, only a unique, primary key value is available for SQL systems. So, a primary key value it is.

What I need is a file or queue to store the tagging information, and which has ... one field, the key value. If I interrogate the file for the key value and get it, the record is tagged. Otherwise, it isn't.

That is, to find out whether or not a given record is tagged, all I need is something like:

```
IsTagged           Function(pLong)

   CODE
   TAG:Ptr = pLong
   If Access:TagFile.Fetch(TAG:PtrKey)
     Return(False)
   Else
     Return(True)
   End
```

and a prototype of:

IsTagged(Long),Byte

If the long is the primary key of both files, IsTagged(CUS:SysID) will tell me whether or not a record is tagged. Further, IsTagged(CUS:SysID) can be used for conditional colors or icons (see Figure 5 above). It can also be used in the SetQueueRecord method to conditionally display something in the browse.

All I need now is a way to:

Set and Remove Tags

Since I am dealing with an external file or queue, setting a tag is just a matter of adding a record. Removing a tag is just a matter of deleting a record. This is totally standard stuff and the only twist is ensuring I don't add a value twice.

```
MakeTag Procedure(pLong)

   CODE
   TAG:Ptr = pLong
   If Access:TagFile.Fetch(TAG:PtrKey)
     Access:TagFile.Insert()
   End
```

If the value passed is not already in the TagFile, add it.

Removing a tag is every bit as hard:

```
DeleteTag       Procedure(pLong)

   CODE
   TAG:Ptr = pLong
   If ~Access:TagFile.Fetch(TAG:PtrKey)
     Relate:TagFile.Delete()
   End
```

If the passed value is in the file, delete it. And, to clear all tags, just loop through the file, deleting records.

The only thing required is to ensure that the highlighted record is in memory (otherwise the wrong key value will be captured). And, that's no problem. Before the tag/untag procedure is called, insert:

```
BRW1.UpdateBuffer
```

or

```
Get(Queue:Browse:1,Choice(?Browse:1))
```

(If you want to use a queue, the function code is at the end of this article.)

File Or Queue?

Queues, of course, disappear when the app terminates (and if I do not Free the queue, the memory it uses also disappears. Queues, therefore, are entirely appropriate if tags only need to last as long as the application is running.

Also, queues are created in the client's workspace. So, queues mean that multiple users can tag records without my having to worry about keeping them separate.

Files are more flexible but a bit more work on account of that flexibility. If I do not want tags to persist from session to session, I can Remove the file. If I want them to persist, I don't Remove it. If I want users tags separated, I create the file on the user's disk, not the network drive. But if tags need to be shared, I can create the file in a shared location.

External Access

Accessing tagged records from other procedures, reports and processes being the most common, is now falling-off-a-log easy:

```
If IsTagged(Cus:SysID)
  Return Record:Filtered
Exit
```

or, simply:

```
IsTagged(Cus:SysID)
```

in the Filter prompt.

Summary

Lots of Clarion users see that marking is available, see a use for it, try to implement it and can’t.

Creating a reliable, consistent and workable substitute really isn’t that hard. What is hard is turning this into a template, which this implementation simply screams for. Luckily, there are several available (absolutely nothing beats not having to write code at all).

Download the demo app

Sample Code For Using Queues

```
IsTagged           Function(pPoint)

   CODE
   TQ_:Ptr = pPoint
   Get(TagQueue,TQ_:Ptr)
   If ~ErrorCode()
     Return(True)
   Else
     Return(False)
   End


MakeTag            Procedure(pPoint)

   CODE
   TQ_:Ptr = pPoint
   Get(TagQueue,TQ_:Ptr)
   If ErrorCode() then Add(TagQueue,TQ_:Ptr).


ClearTag           Procedure(pPoint)
   CODE
   TQ_:Ptr = pPoint
   Get(TagQueue,TQ_:Ptr)
   If ~ErrorCode() then Delete(TagQueue).
```

Steve Parker started his professional life as a Philosopher but now tries to imitate a Clarion developer. A former SCCA competitor, he has been known to adjust other competitor's right side mirrors - while on the track (but only while accelerating). Steve has been writing on Clarion since 1993.