INVEST in your own abilities Clarion magazine

published by CoveComm Inc.

Search

Clarion MAGAZINE

Issue Index March 2000

The Bitlist class and templates
Jeff Slarve's open source BitList template and class are a great way to manage large numbers of True/False values with a minimum of overhead.
(Mar 7,2000)

Advanced Skeletons: Managing Hyperlinks
The older Internet Connect templates have an option called Autospot Hyperlinks. But how do you create hyperlinks in Web Edition?
(Mar 7,2000)

Monthly Source Code Available
Now Clarion Magazine has monthly source code zips available for download along with the monthly PDFs.
(Mar 7,2000)

Next Issue March 23
The next issue of Clarion Magazine will be published Thursday, March 23rd. The normal Tuesday publication schedule resumes next week.
(Mar 14,2000)

Skeleton Bidding And Selection
WebBuilder lets you create a variety of skeletons for a given type of control. How do you know which skeletons will get used for which control, and when? Steve Parker explains all.
(Mar 14,2000)

Supercharged Editing: Using Clarion Macros
The Clarion editor's macro capability isn't exactly complete, but as James Cooke shows it can be a powerful tool nonetheless.
(Mar 14,2000)

Nov/Dec Source Zips Renamed
The November and December source zips have been renamed to avoid name clashes with other source zips.
(Mar 14,2000)

First Impression: Foundations of Clarion 5 Interactive Self-Study CD
Pat O'Brien gets a pleasant surprise when he sits down to the TopSpeed Clarion 5 Interactive Self-Study CD.
(Mar 23,2000)

Adding A Class To Your ABC Program
Not sure how to add a custom class to your ABC app? Tom Ruby explains all, and throws in a serial/parallel port communication class for good measure.
(Mar 23,2000)

The Clarion Advisor: Adding/Removing Key Tabs
Clarion's Application Generator does a great job of creating tabbed browses, with one tab for each key. But what happens when you want to change the order of the tabs on a browse, or add/remove one or more tabs?
(Mar 23,2000)

The Clarion Challenge: Shoot Yourself In The Foot
Write a Clarion version of the classic "Shoot yourself in the foot" code and become famous!
(Mar 23,2000)

Tool Talk: Reviews News
Tom Hebenstreit mulls over TopSpeed's Application Review service, and previews upcoming product reviews.
(Mar 28,2000)

Editorial
TopSpeed's Web Builder looks good for web development, but can the underlying application broker take the heat? Inquiring minds...
(Mar 28,2000)

The Clarion Advisor: Browse Popup Menu Tricks
The Clarion Advisor takes a tour of popup menus, a standard (and handy) feature of Clarion browse boxes.
(Mar 28,2000)

Subclassing With A Twist
Subclassing doesn't always mean OOP - sometimes it means intercepting Windows messages to do things Clarion wouldn't otherwise let you do.
(Mar 28,2000)

March 2000 News
Clarion news, notes, and happenings from around the globe.
(Mar 28,2000)

Clarion MAGAZINE

The BitList Template: A Guided Tour

by Dave Harms

The Bitlist class and templates
(Mar 7,2000)

Advanced Skeletons: Managing Hyperlinks
(Mar 7,2000)

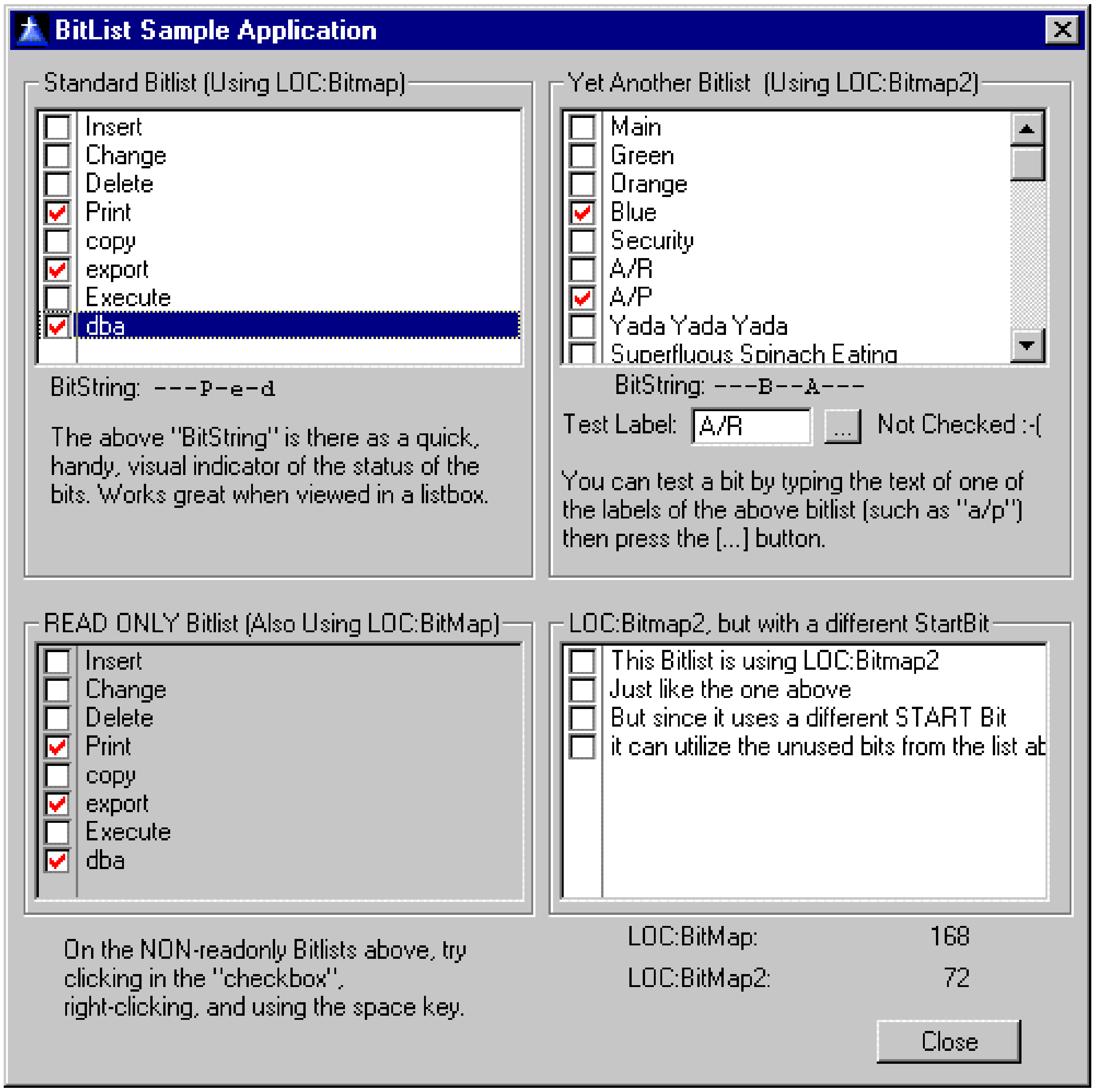
Monthly Source Code Available
(Mar 7,2000)

One of Clarion add-ons available through the Clarion Open Source Project is Jeff Slarve's BitList template. The idea behind BitList is that even in an era when storage is cheap, it sometimes makes sense to store binary values in individual bits rather than wasting whole bytes. The BitList template slices `LONG`s up into 32 separate fields, each of which can be represented by a checkbox in a scrolling list. This may sound a bit tricky, but the template makes it child's play.

Figure 1 shows Jeff's example app with four BitList controls. Three of these controls are read/write, and the check box can be toggled either with the mouse or the spacebar. One of the controls (lower left) has been set to read-only.

Figure 1. The BitList example application.

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



Standard Bitlist (Using LOC:Bitmap)

- [ ] Insert
- [ ] Change
- [ ] Delete
- [x] Print
- [ ] copy
- [x] export
- [ ] Execute
- [x] dba

BitString: `---P-e-d`

The above "BitString" is there as a quick, handy, visual indicator of the status of the bits. Works great when viewed in a listbox.

Yet Another Bitlist (Using LOC:Bitmap2)

- [ ] Main
- [ ] Green
- [ ] Orange
- [x] Blue
- [ ] Security
- [ ] A/R
- [x] A/P
- [ ] Yada Yada Yada
- [ ] Superfluous Spinach Eating

BitString: `---B--A---`

Test Label: A/R ... Not Checked :-(

You can test a bit by typing the text of one of the labels of the above bitlist (such as "a/p") then press the [...] button.

READ ONLY Bitlist (Also Using LOC:BitMap)

- [ ] Insert
- [ ] Change
- [ ] Delete
- [x] Print
- [ ] copy
- [x] export
- [ ] Execute
- [x] dba

LOC:Bitmap2, but with a different StartBit

- [ ] This Bitlist is using LOC:Bitmap2
- [ ] Just like the one above
- [ ] But since it uses a different START Bit
- [ ] it can utilize the unused bits from the list ab

On the NON-readonly Bitlists above, try clicking in the "checkbox", right-clicking, and using the space key.

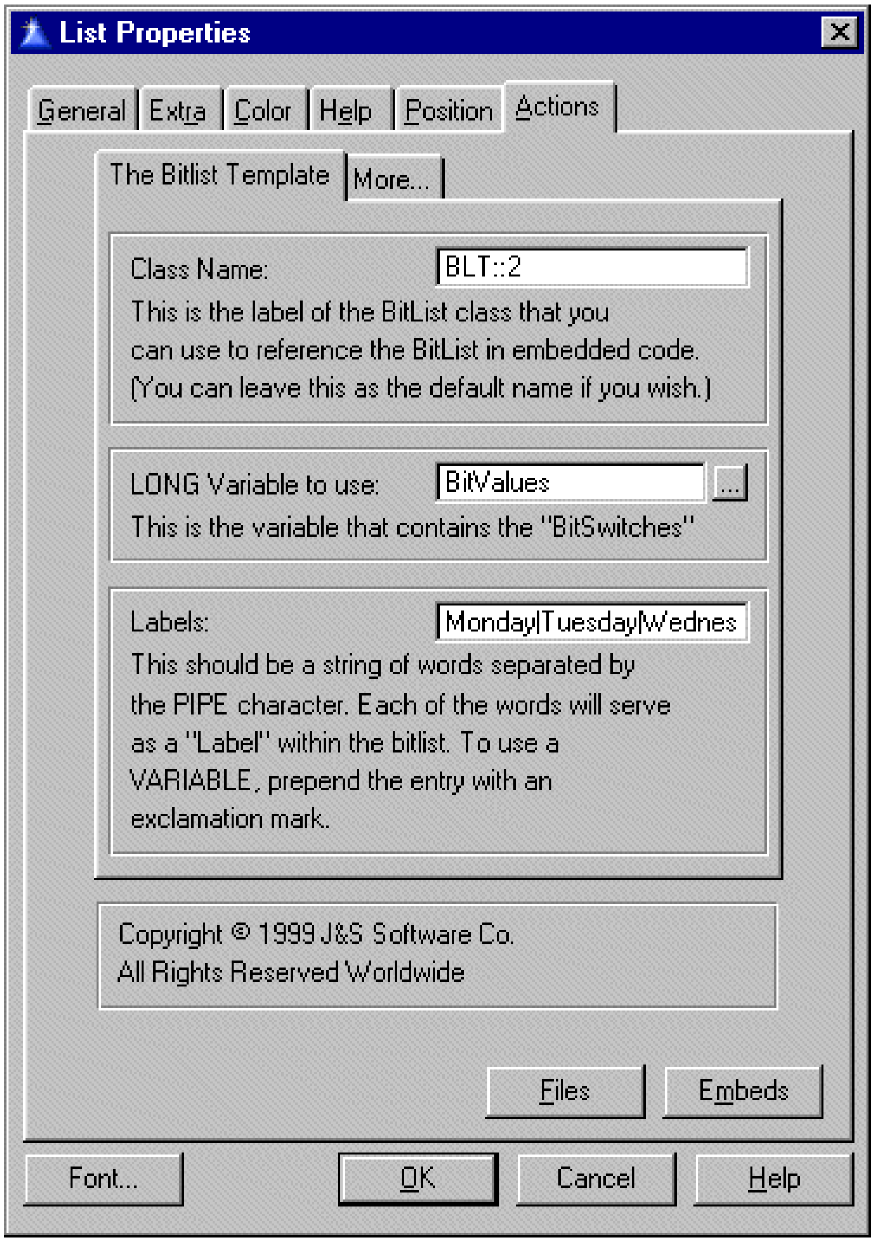
LOC:BitMap:	168
LOC:BitMap2:	72

Close

So how difficult is it to implement one of these controls? Try this. Download the BitList code, and install the BitList.CLW and BitList.INC files in your LIBSRC directory, and the ICO files in your IMAGES directory. Install BitList.TPL in the TEMPLATE directory, and register it.

Now create an app, or use an existing app. Create or use an existing window, and populate the BitListClass BitList control. Right-click on the BitList control and choose Actions, to bring up the Actions tab on the List Properties dialog (see Figure 2).

Figure 2. The BitList template settings.



Class Name: BLT::2
This is the label of the BitList class that you can use to reference the BitList in embedded code. (You can leave this as the default name if you wish.)

LONG Variable to use: BitValues
This is the variable that contains the "BitSwitches"

Labels: Monday|Tuesday|Wednes
This should be a string of words separated by the PIPE character. Each of the words will serve as a "Label" within the bitlist. To use a VARIABLE, prepend the entry with an exclamation mark.

Copyright © 1999 J&S Software Co.
All Rights Reserved Worldwide

You can leave the Class Name field as is or change it to something more descriptive. In the LONG Variable to use field choose an appropriate variable. This will probably be a LONG field in a data file, but could also be a local or global variable if that's appropriate.

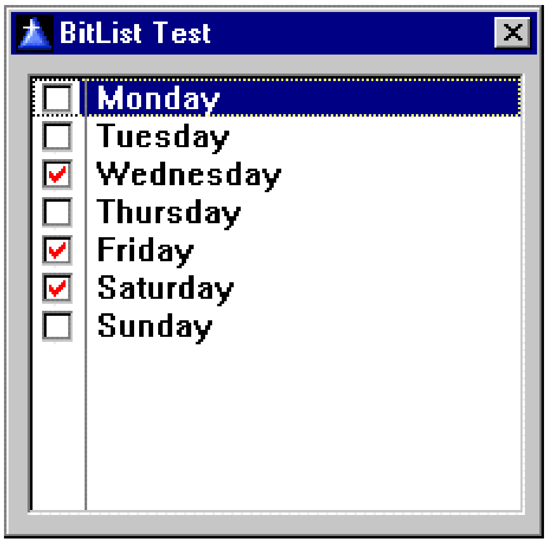
Finally, fill in the values you want to display for each bit, separated by the | character. For example, to display a list of the days of the week, you could enter:

```
Monday|Tuesday|Wednesday|Thursday|Friday|Saturday|Sunday
```

You can also use a variable in this prompt if you prepend it with an exclamation mark, and you may find it easiest to set up global variables for these display values if you have more than one BitList displaying a particular `LONG` field.

Compile and run your application, and you'll have something like Figure 3.

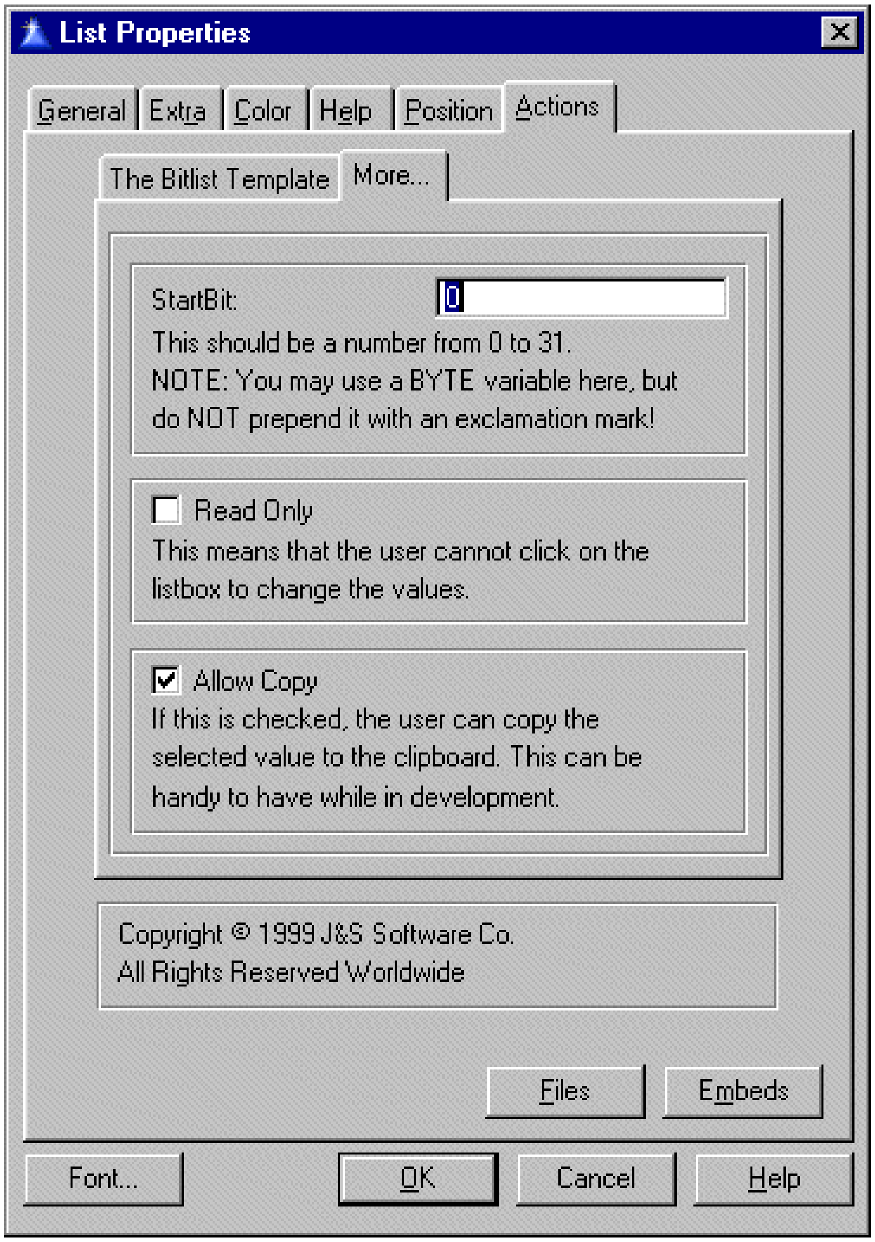
Figure 3. An example BitList control displaying the days of the week.



There are a couple of other options on the template's second tab, as shown in Figure 4. You can specify the start bit, which lets you put several sets of data in the same `LONG` field. In Figure 1, the BitList controls on the right side of the window are displaying data from the same LONG field, but the upper control starts with bit 0 while the lower control starts with bit 15.

You can also set the control to read-only, and allow the user to copy the currently selected bit's text to the clipboard.

Figure 4. Additional BitList template options.



List Properties

General | Extra | Color | Help | Position | Actions

The Bitlist Template | More...

StartBit: 0

This should be a number from 0 to 31.
NOTE: You may use a BYTE variable here, but do NOT prepend it with an exclamation mark!

☐ Read Only
This means that the user cannot click on the listbox to change the values.

☑ Allow Copy
If this is checked, the user can copy the selected value to the clipboard. This can be handy to have while in development.

Copyright © 1999 J&S Software Co.
All Rights Reserved Worldwide

The example application demonstrates another feature of the BitList class: using the `CheckBitStr()` method you can query the value for any particular bit by the associated text. For instance, if your class is called `DateBitList`, calling

```
DateBitList.CheckBitStr(‘Monday’)
```

would return 0 if unchecked, 1 if checked, and 2 if the string ‘Monday’ is invalid.

The BitList template is very handy for storing large numbers of non-keyed true/false values, not just because it economizes on storage space, but because you don’t have to go to the hassle of declaring a bunch of variables either. And unless you're already at your 32 checkbox-per-LONG limit, you can add new fields without creating new variables.

This free product is available under the terms of the Developers Open Source Public License. You can download the BitList template, class, and example app here.

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.*

Clarion MAGAZINE

The Bitlist class and templates
(Mar 7,2000)

Advanced Skeletons: Managing Hyperlinks
(Mar 7,2000)

Monthly Source Code Available
(Mar 7,2000)

Advanced Skeletons: Managing Hyperlinks

by Steve Parker

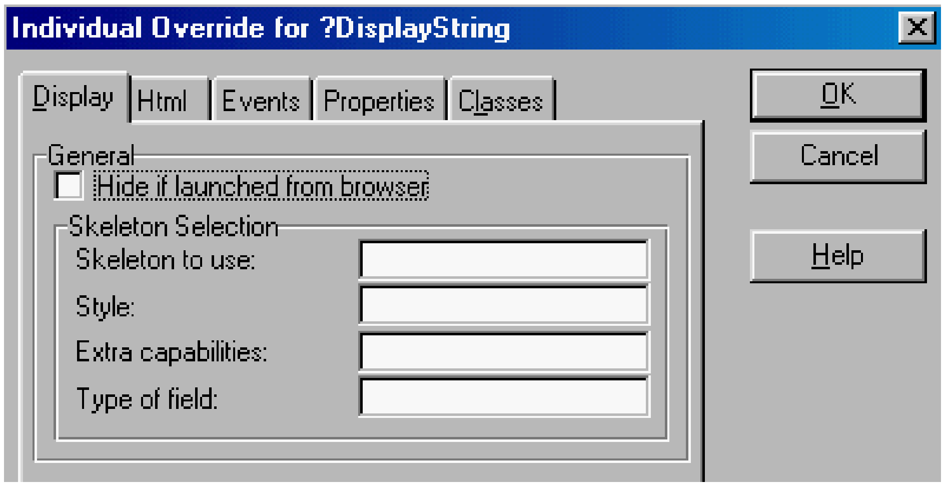
The Internet Connect templates have a control option, "Autospot Hyperlinks." This option is available for both strings and for list boxes. If checked, a string is treated as a hyperlink and any listbox string containing a valid hyperlink prefix also becomes an active hyperlink.

Autospot Hyperlinks seems to have gone missing from the WebBuilder skeletons. So, how does one create hyperlink in 5.5?

The WebBuilder preliminary documentation provides three model skeletons to implement hyperlink and mailto functionality. To use them, you need to create the files in your skeleton directory (if you haven't yet, they are available for download).

Once you have skeleton files stored to disk, assuming the procedure is essentially complete, go to the Procedure Properties window, click Internet Options|Controls and select the control containing (or which will contain) the URL. Click Properties and provide the information necessary for the templates to select the appropriate skeleton (see Figure 1).

Figure 1. Skeleton selection.



General
Hide if launched from browser
Skeleton Selection
Skeleton to use:
Style:
Extra capabilities:
Type of field:

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

Complete the Capabilities and Style prompts with the information culled from the skeleton header (see Figures 2, 3 and 4):

Figure 2. Skeleton capabilities and style.

```
<meta name="ts-control" content="sstring">
<meta name="ts-capabilities" content="hyperlink">
<meta name="ts-style" content="terse">
</head>
<BODY>
<!-- link.string.htm -- Start -->
```

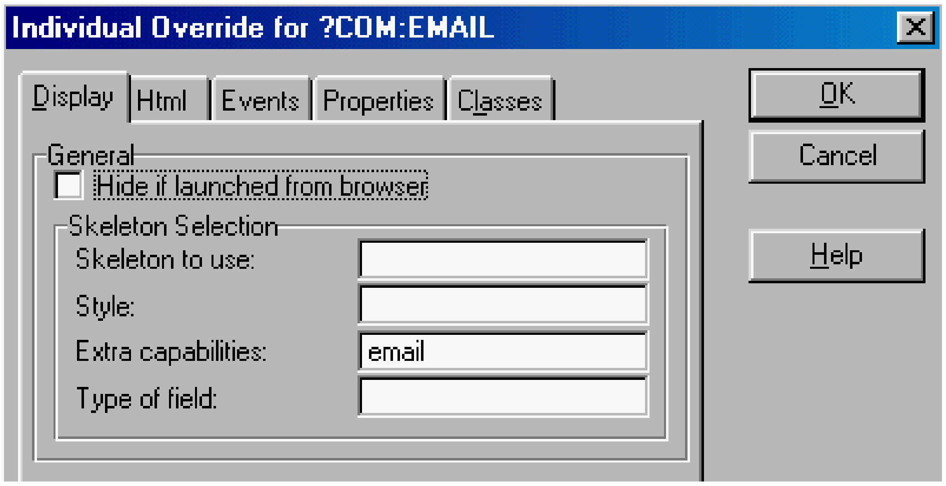
Figure 3. Skeleton alternate capabilities and style.

```
<meta name="ts-control" content="sstring">
<meta name="ts-capabilities" content="hyperlink">
<meta name="ts-style" content="verbose">
</head>
<BODY>
<!-- link.string2.htm -- Start -->
```

Figure 4. Skeleton for email address.

```
<meta name="ts-control" content="sstring">
<meta name="ts-capabilities" content="email">
</head>
<BODY>
<!-- email.string.htm -- Start -->
```

Figure 5. Completed property sheet.



General
Hide if launched from browser
Skeleton Selection
Skeleton to use:
Style:
Extra capabilities: email
Type of field:

Alternately, you can try hard wiring the file name in the first prompt (in these particular cases, hard wiring the skeleton name won't work; using the email.string.htm skeleton in the enclosed sample app, specifying the capabilities did and all three of these skeletons must be handled similarly, as in Figure 5).

As in Internet Connect, the string to which you attach one of these skeletons must contain a complete and correct URL (Internet Address) or IP address. The templates rely on your knowing what you are doing. If you say "This is a hyperlink" by specifying one of these skeletons, the template will take you at your word and create the link code. If the address is no good….

However, the individual fields in a list box are not available in the controls list. This means that none of these skeletons can be attached to a file field in the list. (Internet Connect's Autospot Hyperlinks will find the hyperlink within a listbox string but the HTML parser in WebBuilder does not – `TSSCRIPT` tags and attributes are provided to allow you to create skeletons that contain HTML, rather than simply generate it.)

For that matter, my attempts to custom code the entire linking string, for example:

```
DisplayString = '<<a href="mailto:' & |
 Clip(COM:EMAIL) & '>' & Clip(COM:NAME) & '<</a>
'
```

and populate `DisplayString` in the list were unsuccessful.

However, deleting `DisplayString` (or the file field) from the list, without repopulating it on the window, and then using the code:

```
Target.Writeln(DisplayString)
```

in the list box's After Generating HTML embed will work.

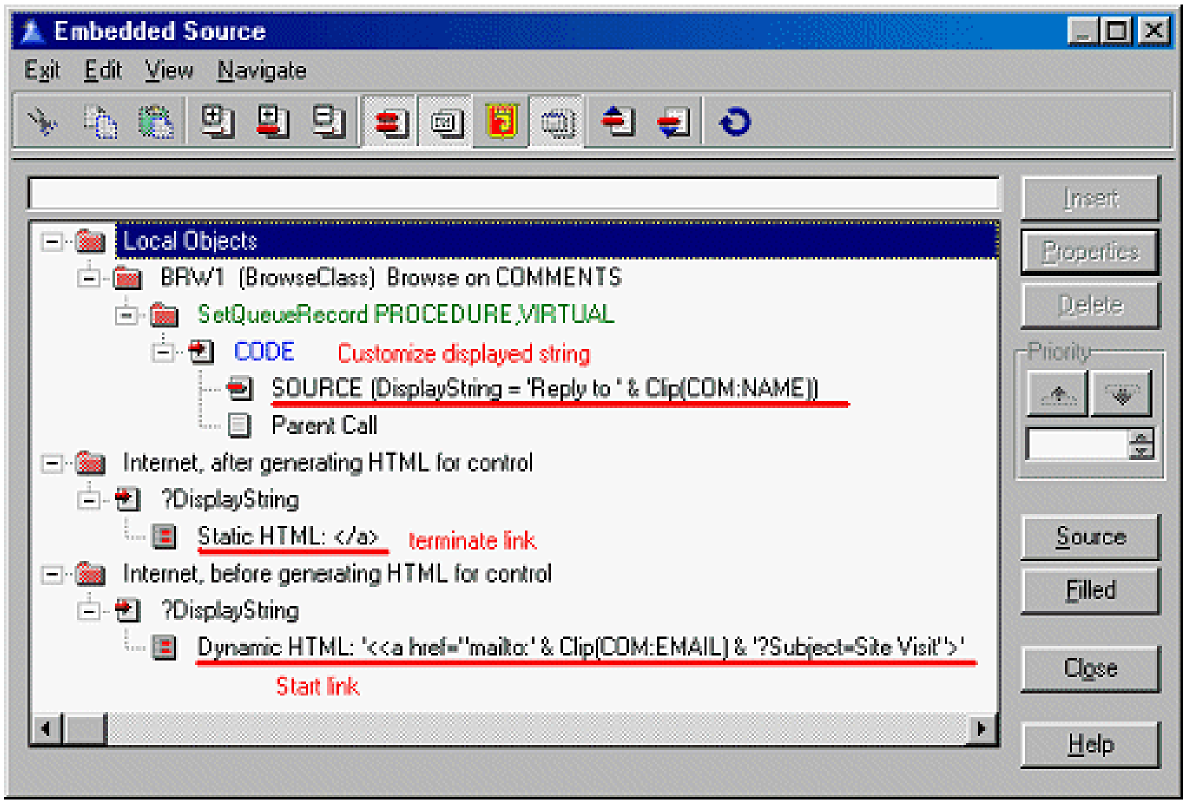
Alternately, moving the string out of the listbox and onto the window (or using a field, file or local variable or string literal) and putting:

```
'<<a href="mailto:' & Clip(COM:EMAIL) & '">'
```

into a Dynamic HTML Code Template before the variable and `'</a>`' in a Static HTML Code Template afterwards works fine (see Figure 6).

The latter method has the virtue of leaving a reminder (the variable) on the window in design mode.

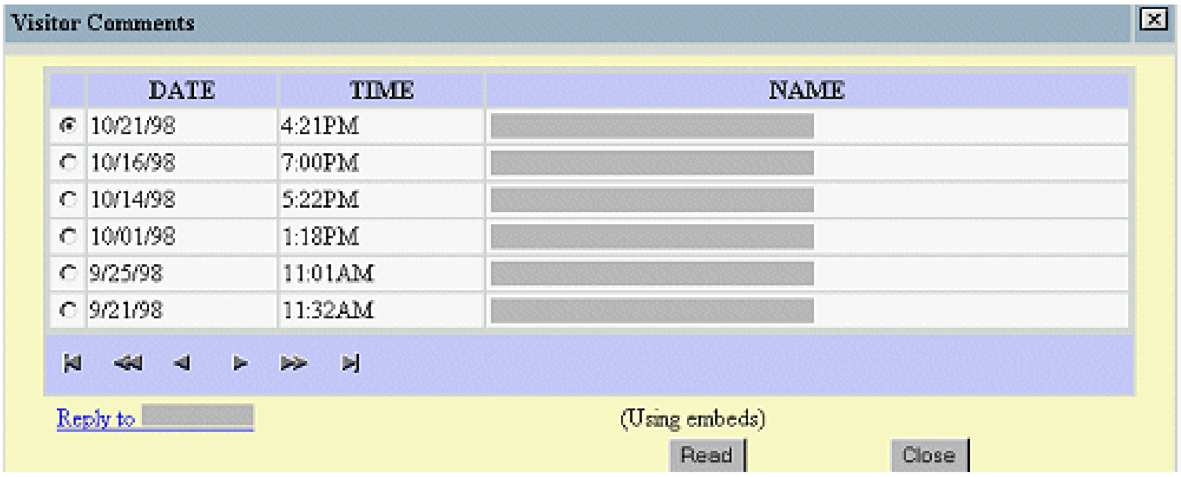
Figure 6. Embeds surrounding a field containing an email address.



```
Local Objects
  BRW1 (BrowseClass) Browse on COMMENTS
    SetQueueRecord PROCEDURE,VIRTUAL
      CODE   Customize displayed string
        SOURCE (DisplayString = 'Reply to ' & Clip(COM:NAME))
        Parent Call
Internet, after generating HTML for control
  ?DisplayString
    Static HTML: </a>   terminate link
Internet, before generating HTML for control
  ?DisplayString
    Dynamic HTML: '<<a href="mailto:' & Clip(COM:EMAIL) & '?Subject=Site Visit">'
              Start link
```

So I have my link (with variations). I just can't have it *in* the list (well, I can, but either I have to use Java listboxes or hand code the entire list and *that's* a whole 'nother story and technique or create a specialized skeleton). In fact, not only does a string placed on the page work, it is dynamically customizable (which I find *very* nice). So, if this is a loss in flexibility from Internet Connect to WebBuilder, it is a small one.

Figure 7. The application in action (some real data has been hidden to protect privacy).



	DATE	TIME	NAME
●	10/21/98	4:21PM	[redacted]
○	10/16/98	7:00PM	[redacted]
○	10/14/98	5:22PM	[redacted]
○	10/01/98	1:18PM	[redacted]
○	9/25/98	11:01AM	[redacted]
○	9/21/98	11:32AM	[redacted]

Reply to [redacted]   (Using embeds)

While listbox text cannot be activated as a hyperlink or, at least, not without substantial additional work (i.e., a specialized skeleton and/or hand coding the list), a customizable "go to" text field is quite easily implemented.

To review:

> **Step 1:** Place a string or variable on the window.
>
> **Step 2:** Place all of the HTML up to but excluding what is to display on the page in a Dynamic HTML Code Template in the string's Before Generating

HTML embed

Step 3: In the string's After Generating HTML embed, finish the link with "</a>" in a Static HTML Code Template

Alternately, it is possible to create the entire HTML string and Target.Writeln the whole thing.

Button, Button, Who's Got the Button?

Another obvious solution is a "go to" or "launch" button or so it seems to those who feel that would be more webbish[1]. Such a button would activate a (dynamically created) hyperlink. Supposedly the link would be contained in a hot field from the file or constructed from file fields and held in a local variable.

Clarion Developers have been asking for this since CWIC beta 1.

Unfortunately, HTML does not support this sort of thing. HTML buttons cannot be hyperlinks (try it, you can't do it). [2]

There are exactly two built-in HTML button objects and, therefore, only two kinds of buttons WebBuilder can generate using HTML: Reset and Submit. Reset clears all the fields on the form and Submit sends the form to the URL/handler specified in the form's ACTION attribute (I suspect that all Clarion buttons are Submits). Any other button requires a custom handler (i.e., Java or Javascript).

The short take? HTML Buttons cannot on their own act as hyperlinks or trigger them. Almost any button you see on a web site serving as a link is actually an image.

Now, a button link *can* be done with Javascript or with Java. Any real button (i.e., non-image) you see on a web site that acts as a link does it through Javascript or Java.

ButtonTest.htm, attached, is a button using Javascript that will trigger a link. If you examine the code (it is adapted from a script I found at CNET.NET for a somewhat different purpose, there weren't any simple "button link" type scripts), you will see that you can dynamically generate this code so that it uses a file field or combination of file fields.

On the other hand, you have to use an HTML button, not a Clarion button in this script. Why? A Clarion control has Before and After Generating HTML embeds but you need an embed inside the button to specify and prime the `onClick()` handler and there is no such embed.

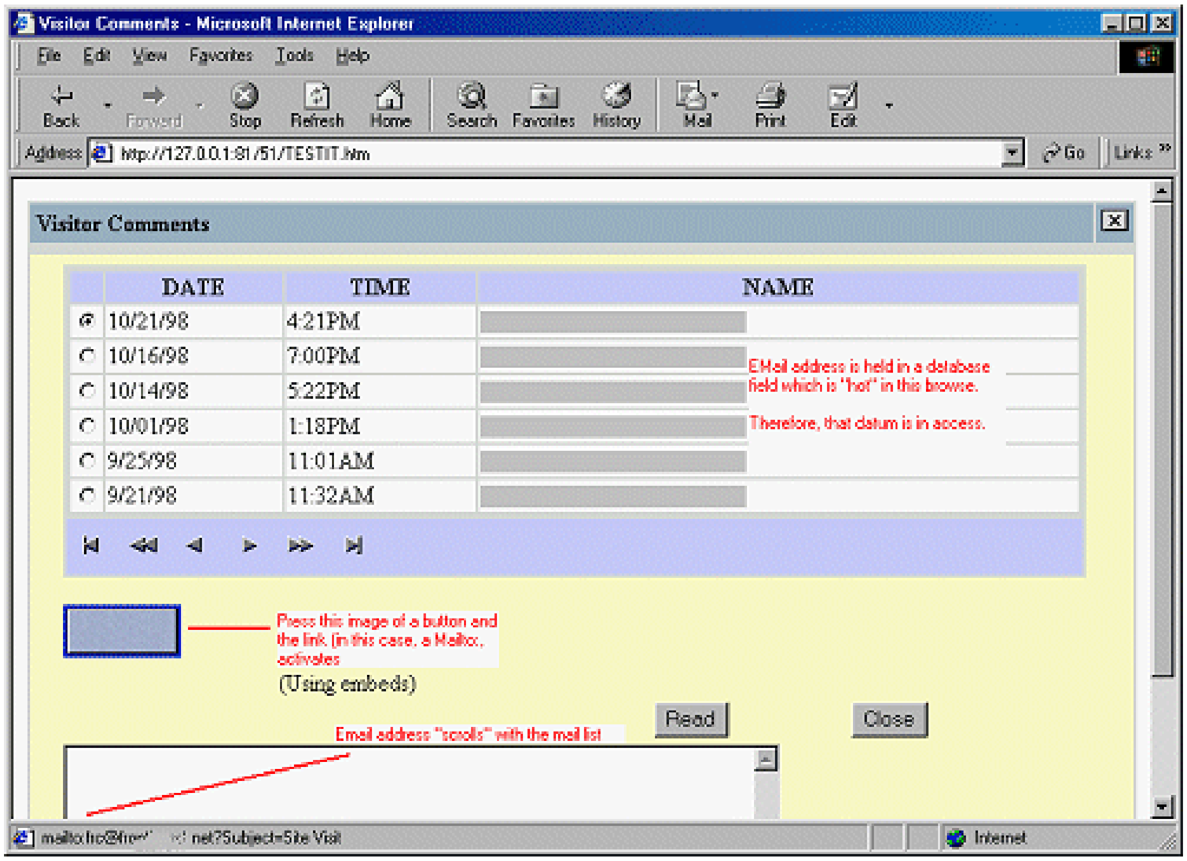
In short, flog this if you wish (but publish the results when you succeed – and you will – that's the deal here) but it's too much like work for me. Too much like work because exactly same result can be accomplished with an image of a button and standard HTML code (and much less of it, too):

Figure 8. Using an image as a link.

```
Target.Writeln('<<br><<a href="mailto:' & Clip(COM:EMAIL)
|
 & '?Subject=Site Visit"><<img border="0" src=' |
 & '"/Blue Blank.gif"><</a>')
```

This is what it looks like:

Figure 9. The running application.


Visitor Comments - 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/TESTIT.htm
Go Links
Visitor Comments
Read
Close
Internet

	DATE	TIME	NAME
●	10/21/98	4:21PM	
○	10/16/98	7:00PM	
○	10/14/98	5:22PM	
○	10/01/98	1:18PM	
○	9/25/98	11:01AM	
○	9/21/98	11:32AM	

EMail address is held in a database field which is "hot" in this browse.

Therefore, that datum is in access.

Press this image of a button and the link (in this case, a Mailto:, activates

(Using embeds)

Email address "scrolls" with the mail list

mailto:hc@fre··· ··net?Subject=Site Visit

and it works quite nicely (Ok, I could have put some text on the button but I just grabbed the first button I found – deadlines, you understand).

What is really very convenient, whether an image or a string is used, is that I only have to type the HTML. The data fields containing the link information, having been added to the queue (by placing them in the list or in the hot list), are always current in memory. I.e., no further code to fetch data is necessary once the list item has been selected.

I see only two reasons to use Javascript. One reason would be to provide substitute text in the status bar when there is a reason to hide the standard link text (viz., the URL). The other is when there is a need to open the link in a second browser window (I do both at CWICWEB's download site).

When Javascript Is Necessary

Javascript is necessary if the link is to be opened in a second window or if the status line text is to be hidden (i.e., the URL usually shows but, if it needs to be hidden under alternate text).

To use a second window and custom status text, adapt the following:

Figure 10. A second window with custom status text.

```
<a HREF="JavaScript:void(0)" onClick="window.open↵
  ('http://localhost','','toolbar=no,directories=↵
  no,captionbar=no,status=yes,menubar=no,scrollbars=yes↵
  ,location=no,width=550,height=400,resizable=yes');↵
  " onmouseover="self.status= 'Check it out.'; ↵
  return true ">Go there now</a>
```

> **NOTE:** Line continuation does not work when dynamically creating these scripts. This is all one line. Substitute the variable containing the URL for "localhost" and the desired status text for "Check it out."

Since large portions of each script will be re-used, it is probably wise to break each script into a series of segments and simply hardcode them into local variables. This makes the code much easier to follow (in the actual code, this is a single line):

```
DisplayString = SEG1 & Clip(FQ:URLPath) & Clip(FQ:FileName) |
 & Clip(FQ:FileType) & SEG2 & Clip(FQ:Description) & SEG3
```

In this case, the `SEG1` variable would contain everything up to the URL. `SEG2` contains everything between the URL and the status text and `SEG3` everything after the status text.

In practice, the "Go there now" text in Figure 10 (which is what is displayed on the page) can be either a string literal or variable, as shown earlier. It can also be a image of a button, using standard HTML, as shown above.

The Point Of The Exercise

With the Internet Connect templates, Autospot Hyperlinks can be turned on for strings or for lists. With WebBuilder, there are three skeletons.

What's the point of demonstrating how to do links in code?

The simple answer is the same as it's always been: flexibility.

All of the supplied methods hard wire what the user sees. In all cases, the user sees the actual URL both on the page and in the status bar (except using the link.string.htm terse-style skeleton which displays a fixed string on the page). Using the method presented here, through the Dynamic HTML Code Template, *I* control what the user sees.

Also, the supplied methods do not readily permit use of any kind of button. The techniques presented here do allow this. It isn't a Clarion button, to be sure, but a button nonetheless.

Lastly, the TopSpeed-supplied methods will activate the link in the current window, leaving your app running on the server and inviting the user to fail to close it properly (not that most do anyway) and displays the target URL in the browser status bar. The presented methods give me the choice to show or not show, to use a new window or not.

Summary

Can you use a button for a hyperlink? Yes.

It will not be a Clarion button; it might be a Javascript button; it might be an image of a button (prettier than the real thing, too). But it can be done.

Could I skeleton-ize the flexibility I want? Probably (for strings, at least). I suspect that by `INCLUDE`ing another, string skeleton, within the link skeleton, to handle the display, this could be done. It would mean specifying alternate skeletons for two controls and I suspect, given the number of times I need to do this, it would not gain me much efficiency.

The demonstration app – for reading a guest book, by the way – shows both the successful techniques as well as the unsuccessful.

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.

Notes

1 Originally coined as "web-ish," I have decided that this really does deserve to be a full fledged adjective. It rhymes with "nebbish;" look it up, you'll get it.

2 It has been pointed out to me that a button can be used as a button in standard HTML using a GET action:

```
<form method="GET" ↵
  action="http://www.clarionmag.com/test/index.html">
  <p><input type="submit" value="Done" name="Test"></p>
</form>
```

Note, however, that the button is not the link. In this case, the form is the link (and this is standard HTML). In other words, this does not appear to add anything to what I say here.

It would be entirely possible (and, on inspection, not at all difficult) to use the techniques described here to implement this code, including using a variable for the URL. It might even be possible to use a Clarion button.

Sound like a candidate for a Control Template.

published by CoveComm Inc.

Clarion MAGAZINE

Clarion Magazine Information
December, 1999

Publishing Schedule

Clarion Magazine is now published four times per month, usually (but not always) on Tuesdays. This allows us to present you with up-to-date news items as well as a steady diet of Clarion programming information from the best writers in the Clarion community. As some months have five Tuesdays, publication occasionally skips a week.

In other respects, Clarion Magazine resembles a monthly magazine. Internally, the contents of the magazine are organized by month, and this is reflected in the Site Index. When you purchase a year's subscription, you have access to all of the articles for the month in which you begin your subscription, and for the next 11 months.

Clarion Magazine Main Page
(Feb 12,1999)

The Clarion Magazine FAQ
(Feb 24,1999)

The Site Index
(Feb 12,1999)

Subscribe to Clarion Magazine!
(Feb 8,1999)

The Subscription Agreement
(Feb 5,1999)

Refund Policy
(Feb 8,1999)

Subscriber Services
(Feb 8,1999)

Information For Authors
(Feb 2,1999)

How To Contact Clarion Magazine
(Feb 2,1999)

The Unofficial Clarion OOP Page
(Feb 6,1999)

Lend Us Your Links
(Feb 7,1999)

Getting Access To Clarion Magazine
(Feb 7,1999)

Clarion Magazine's Automated Mailing Lists
(Jan 31,1999)

Clarion Magazine Is Best Read With Verdana
(Feb 7,1999)

Clarion Links By Category
(May 17,1999)

Free Newsgroups For Clarion User Groups

(May 10,1999)

Free Utilities: Class Browser And Compile Manager
(Dec 7,1999)

Alphabetical Author Index
(Jan 4,2000)

Alphabetical Article Index
(Jan 4,2000)

Clarion's Publication Schedule
(Feb 5,1999)

Advertising Policy
(Feb 7,1999)

Open Source Code Available Now
(Mar 8,1999)

The Masthead
(Feb 15,1999)

Clarion MAGAZINE

Advanced Skeletons: Skeleton Bidding and Selection

Next Issue March 23
(Mar 14,2000)

Skeleton Bidding And Selection
(Mar 14,2000)

Supercharged Editing: Using Clarion Macros
(Mar 14,2000)

Nov/Dec Source Zips Renamed
(Mar 14,2000)

by Steve Parker

WebBuilder uses a combination of skeleton HTML and embedded HTML to create web pages. Sounds simple enough. But you can use multiple versions of a given skeleton in a single app and you may even want to specify a particular skeleton at run time, based on a condition, perhaps.

How do you know/decide which skeletons get used, and when? How are skeletons chosen when you don't specify one?

The only thing that the WebBuilder extension insists on is knowing where the skeletons can be found (Figure 1). By default, they are expected in a directory, SKELETON, below the directory containing the .EXE.

So,

```
C:\PAR2\EXEC\COOLFAQS.EXE
```

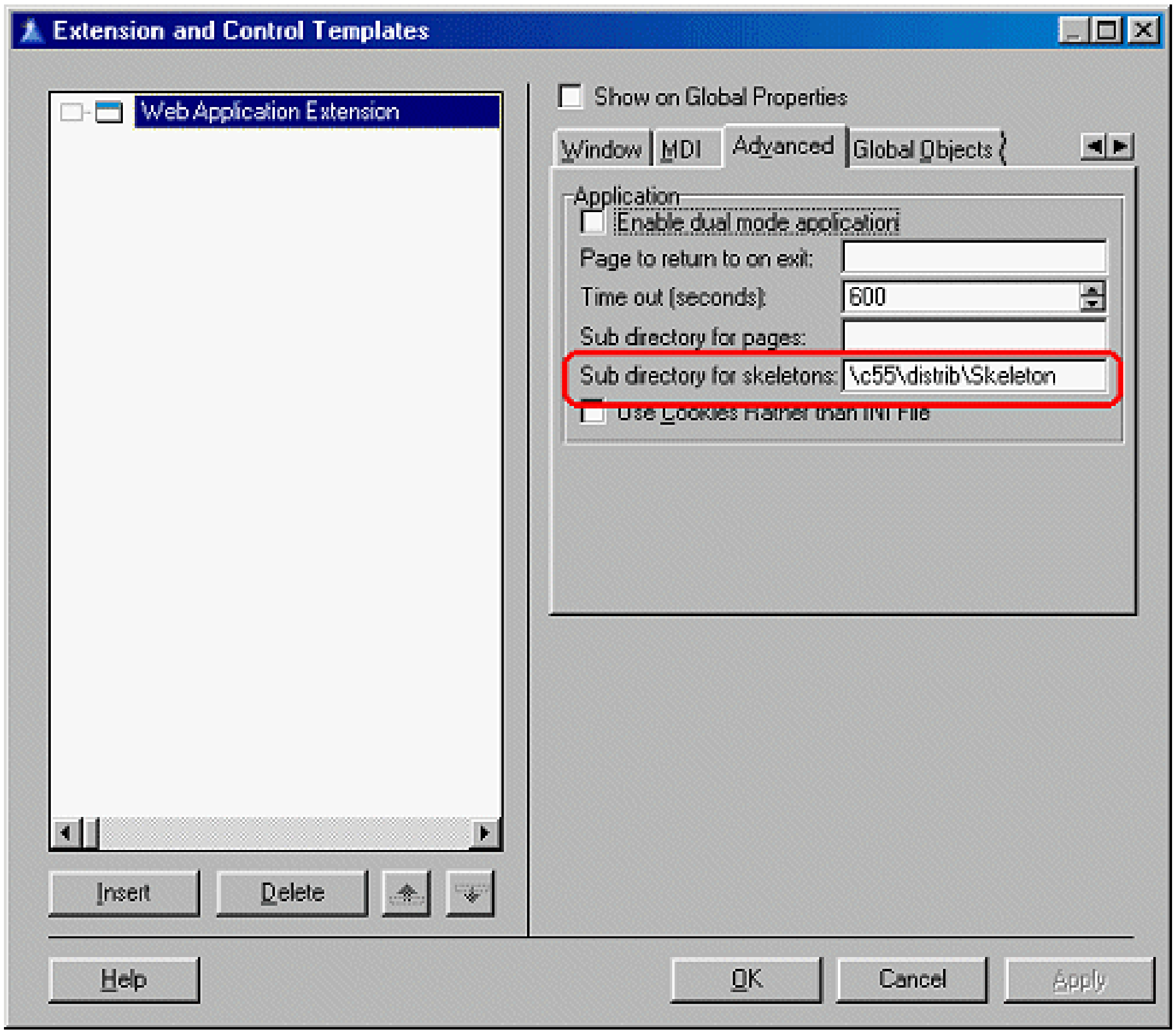
expects to find the skeleton files in

```
C:\PAR2\EXEC\SKELETON
```

However, any directory can be used (Figure 1, again), so long as the templates know which.

Figure 1. Specifying a skeleton subdirectory.

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


Extension and Control Templates

Web Application Extension

Show on Global Properties

Window | MDI | Advanced | Global Objects

Application

Enable dual mode application

Page to return to on exit:

Time out (seconds): 600

Sub directory for pages:

Sub directory for skeletons: \c55\distrib\Skeleton

Use Cookies Rather than INI File

So, WebBuilder now knows where to find the skeletons. But how does it know which one to use for a given control?

I haven’t the vaguest idea of how controls are matched to skeletons (in other words, everything that follows is retroduction – educated guesswork). But there are enough clues as to what must be happening that some educated guesses are possible.

But First …

There is a basic fact that bears repeating. With the exception of Window.HTM (that’s in beta 2; in beta 1, Window.HTM also calls WinCore.HTM, ColorA and ColorB.HTM), a skeleton controls the production of HTML for a *single* control.

This has some very important implications. Specifically, when non-standard HTML is required, a different skeleton is also required, one skeleton for each type of control. But, and a few Clarion developers have had difficulty with this, the alternate skeleton must be "nominated" (meaning to be named explicitly) for a specific window control. To make a field an email control (using the `email.string.htm` skeleton in the documentation), the skeleton is attached to the control. It is not attached to a variable; it is not passed parameters. There must be a control, on the window, containing an email address and the app is instructed to use this skeleton for this control.

In this case, the option is hard wired: a specific skeleton is explicitly named. But other options are active without coding (such as "normal" controls for which no skeleton is nominated at all).

The Clues: Hard Wired Options

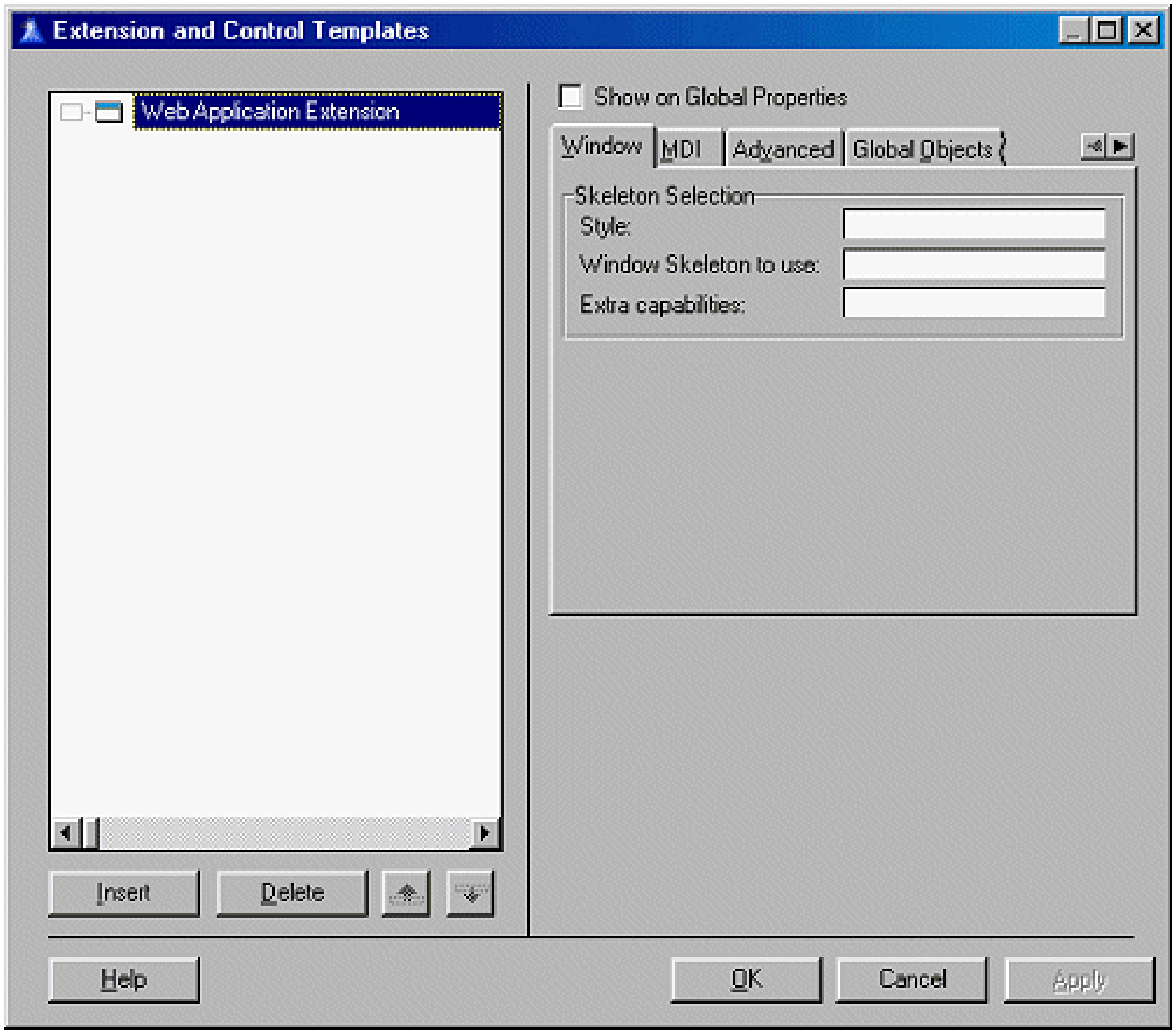
At the application global level, you find the following options:

```
Style
Window Skeleton
```

and

```
Extra Capabilities
```

Figure 2. Global skeleton options.



Extension and Control Templates

Web Application Extension

Show on Global Properties

Window | MDI | Advanced | Global Objects

Skeleton Selection

Style:	
Window Skeleton to use:	
Extra capabilities:	

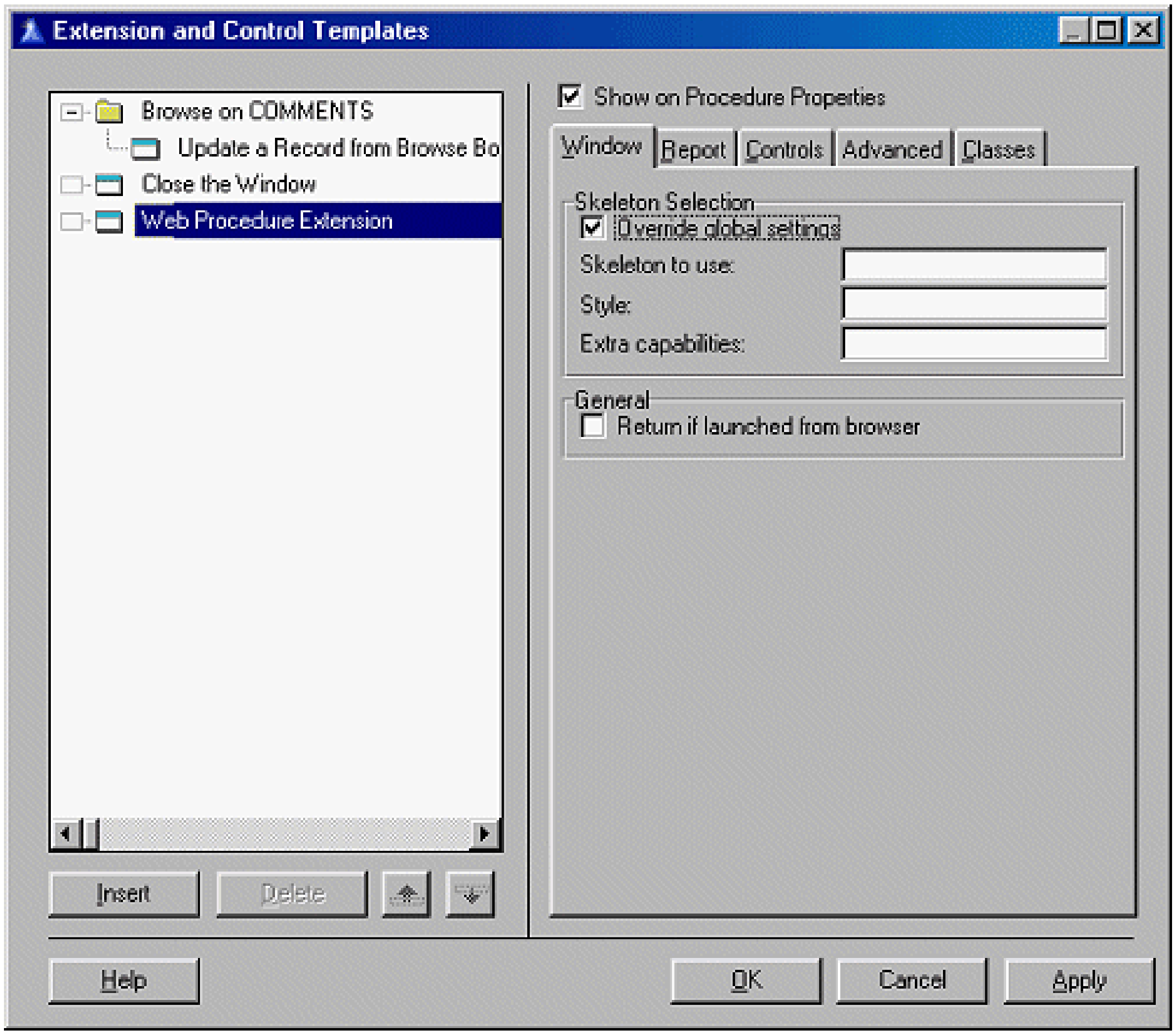
And, at the individual control level, are these options:

```
Skeleton to Use
Style ("Theme" in B2)
```

and

```
Extra Capabilities ("Type of field" in beta 2)
```

Figure 3. Control skeleton options.



☑ Show on Procedure Properties

Skeleton Selection
☑ Override global settings
Skeleton to use:
Style:
Extra capabilities:

General
☐ Return if launched from browser

If I name a file in `Window Skeleton` or `Skeleton to Use`, other options become unavailable. These options are hard wired, totally determinative of the file to use, making others irrelevant.

`Window Skeleton`, as show in previous articles in this series, hard codes a skeleton which determines the look and feel of all pages for the application. Window.HTM, or its proxy, controls the color scheme and background images for all pages. (All customizations discussed in previous articles impacting WinCore.HTM or ColorA.HTM will, in beta 2, need to be moved into Window.HTM.)

`Skeleton to Use` tells the HTML parser to use a specific file for a specific control. (To change a control type globally, either change the base skeleton or name and create a style in the global prompt.)

The Bidding System

If these options by-pass the normal selection of skeletons, entries in the "Style" and "Capabilities" prompts must have similar effect also. And, as you can find by examining the supplied skeletons, there are three TSSCRIPT variables that can receive developer-specified values:

```
ts-control
ts-style
```

and

```
  ts-capabilities
```

What do these mean?

It seems clear that "ts-control" means "Topspeed control type." With this in mind, an examination of these three skeleton code snippets (taken from the link and email skeletons in the initial WebBuilder documentation) suggests that four different skeletons produce a single type of control:

Figure 4. The link.string.htm skeleton.

```
<meta name="ts-control" content="sstring">
<meta name="ts-capabilities" content="hyperlink">
<meta name="ts-style" content="terse">
</head>
<BODY>
<!-- link.string.htm -- Start -->
```

Figure 5. The link.string2.htm skeleton.

```
<meta name="ts-control" content="sstring">
<meta name="ts-capabilities" content="hyperlink">
<meta name="ts-style" content="verbose">
</head>
<BODY>
<!-- link.string2.htm -- Start -->
```

Figure 6. The email.string.htm skeleton.

```
<meta name="ts-control" content="sstring">
<meta name="ts-capabilities" content="email">
</head>
<BODY>
<!-- email.string.htm -- Start -->
```

Four? You forgot SString.HTM, the parent of these controls, didn’t you?

This is where things get a bit interesting. If I name, for example, `email.string.htm` in the `Skeleton to Use` prompt for a string, it will not turn into an email link as expected. But, if I type "email" in the `Extra Capabilities` prompt (matching the "ts-capabilities" variable in `email.string.htm`), it does.

Obviously, the `Extra Capabilities` prompt (the `Style` prompt, as well) somehow assists the parser determining which of several files to use.

Here is what I think is happening.

At runtime, when an app is started, the skeleton directory is interrogated and a queue is created. This queue appears to contain four entries: *control type*, *capabilities*, *style* and, of course, *file name*. So, the entries for the skeletons above might look like:

Control	Capabilities	Style	File

SString			.\skeleton\sstring.htm
SString	Email		.\skeleton\email.string.htm
SString	Hyperlink	terse	.\skeleton\link.string.htm
SString	Hyperlink	verbose	.\skeleton\link.string2.htm

> **NOTE:** I have not specifically tested and, therefore, am not certain of the order of Capabilities and Style. These entries could be reversed but the wording in the last two skeletons does indicate that Capability is a higher node than Style.

Here's the important part: skeleton selection *appears* to operate as if the first three entries are primed (from the templates) and then:

```
Get(SkeletonQueue,+content,+capabilities,+style)
```

The first entry matching wins and the file it points to is the one used. Among other things, "file" is not a sort node and that is why "email.string.htm" did not work above. The file name is picked up only if, at least, control has a value.

Implications

Two very important points emerge from this.

First, to nominate an alternate skeleton, you cannot simply name a new file. The contents of the control parameter must be changed (or values given in the capabilities or style prompts). And, while I have simply changed the contents of the control parameter successfully (hoping that the HTML parser was doing an `INSTRING` and would pickup the data needed), this may result in a control not displaying as expected due to a failure to match up to the expected skeleton file.

Second, it is possible to nominate an entire series of alternate skeletons in a single go. By making an entry in the global "Style" prompt, for example, all skeletons including that style will be used in preference to any other skeletons. Cool.

Further (and my thanks to Jim DeFabia for pointing this out to me), the parser will walk the directory tree when creating this queue.

That is, suppose I create a directory below the main skeleton directory to store a skeleton (or skeleton set) with a different look and feel. If I then create one or more skeletons in that directory and give them a unique style or capability, they will be found (and, do notice that I did *not* say that the new file required a new file name, if it is a different directory, it doesn't). Any skeletons required but not in this new directory will still be found; they are in the top level skeleton directory.

So, placing my personalized skeleton(s) in a separate directory, might result in a bidding queue like:

Control	**Capabilities**	**Style**	**File**
SString			.\skeleton\sstring.htm
SString		**shp**	.\skeleton\shp\sstring.htm

SString	email		.\skeleton\email.string.htm
SString	hyperlink	Terse	.\skeleton\link.string.htm
SString	hyperlink	Verbose	.\skeleton\link.string2.htm

Way cool. Yes, indeed, I can have multiple files in multiple subdirectories, sharing a common file name but with different style settings. I can nominate simply by setting the style.

Runtime Selection

Since an entry in a template prompt can change the skeleton used at runtime, it is obvious that there is a WebBuilder method that can be called to implement this.

Subtle stratagem time again. Make an entry in the `Skeleton to Use` prompt and look at the generated source:

Figure 7. The generated GetSkeletonAttr method.

```
Web:List:2.GetSkeletonAttr PROCEDURE(SIGNED whichAttr)

ReturnValue          ANY

  CODE
  ReturnValue = PARENT.GetSkeletonAttr(whichAttr)
  ! [Priority 6000]
  CASE whichAttr
  OF SkeletonAttr:Name
    ReturnValue = 'shptable3.htm'
  END
  RETURN ReturnValue
```

And, there it is: `GetSkeletonAttr`. (To get the object name, right click the control and press "Embeds." The WebBuilder object name will be in the list below the Local Objects name.)

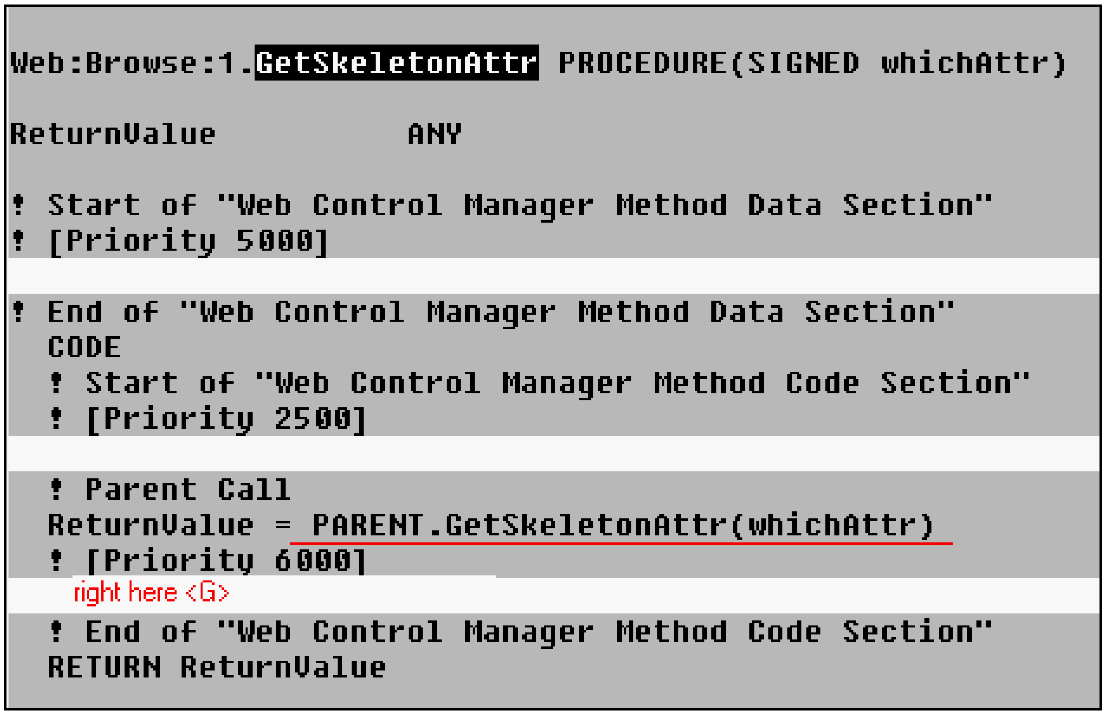
Doing the same for the remaining prompts (remember, because `Skeleton to Use` disables the remaining prompts, this must be done in at least two steps) shows:

Figure 8. GetSkeletonAttr with the remaining prompts filled in.

```
Web:List:2.GetSkeletonAttr PROCEDURE(SIGNED whichAttr)

ReturnValue          ANY

  CODE
  ! Parent Call
  ReturnValue = PARENT.GetSkeletonAttr(whichAttr)
  ! [Priority 6000]
  CASE whichAttr
  OF SkeletonAttr:Capabilities
    ReturnValue = ReturnValue & ',shpCapability'
  OF SkeletonAttr:Style
    ReturnValue = 'shpStyle'
  OF SkeletonAttr:Type
    ReturnValue = 'shpType'
  END
  RETURN ReturnValue
```

Conveniently - this is all generated code - there is an embed in `GetSkeletonAttr` right after the parent call:

Figure 9. The generated GetSkeletonAttr code in the embed editor.



```
Web:Browse:1.GetSkeletonAttr PROCEDURE(SIGNED whichAttr)

ReturnValue          ANY

! Start of "Web Control Manager Method Data Section"
! [Priority 5000]

! End of "Web Control Manager Method Data Section"
  CODE
  ! Start of "Web Control Manager Method Code Section"
  ! [Priority 2500]

  ! Parent Call
  ReturnValue = PARENT.GetSkeletonAttr(whichAttr)
  ! [Priority 6000]
   right here <G>
  ! End of "Web Control Manager Method Code Section"
  RETURN ReturnValue
```

This makes runtime selection, without using the template prompts, quite straightforward by simply emulating what the templates do (see Figure 10).

Figure 10. Code to insert in the embed point.

```
Case whichAttr
Of SkeletonAttr:Style
  Execute (Today() % 7) + 1
    ReturnValue = ‘SundayStyle’
    ReturnValue = ‘MondayStyle’
    ReturnValue = ‘TuesdayStyle’
    ReturnValue = ‘WednesdayStyle’
    ReturnValue = ‘ThursdayStyle’
    ReturnValue = ‘FridayStyle’
    ReturnValue = ‘SaturdayStyle’
  End
End
```

The only requirement for this to work, ridiculous as it looks, is the line

`<meta name="ts-style" content="SundayStyle">`

in a version of the chosen skeleton, and

`<meta name="ts-style" content="MondayStyle">`

in another version and so on for each of seven skeleton files (probably seven versions of Window.HTM). Again, odd as this looks, using a developer-controlled variable to determine which skeleton is used is easily accomplished.

And, remember, each of these can be in a separate subdirectory below the distributed skeletons and need only contain the files that differ from the stock files.

Summary

I am not certain that a bidding queue, such as that described, is actually created. But the fact is that WebBuilder behaves as if a structure like this is used in the way described.

And, even if nothing remotely like this happens, it did lead me to the `GetSkeletonAttr` method. With `GetSkeletonAttr`, I can make a runtime assignment of skeletons based on a testable condition.

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.

Next Issue March 23
(Mar 14,2000)

Skeleton Bidding And Selection
(Mar 14,2000)

Supercharged Editing: Using Clarion Macros
(Mar 14,2000)

Nov/Dec Source Zips Renamed
(Mar 14,2000)

Supercharged Editing: Using Clarion Macros

by James Cooke

It has been said that one of the requirements for being a good programmer is that the programmer should be inherently lazy...but strategically so. In other words, good programmers minimize future effort by optimizing current effort. A classic example of this is Clarion's screen designer: the programmer has to go through a separate learning curve in order to speed up what he can already do by typing alone.

Programmers have all sorts of tools to make things quicker - if/execute/case structures shorten statements, iteration statements (loops) handle repetitive tasks, templates and OOP produce accurate code quickly. So nobody can deny that programmers are in the business of doing more for less. And this is the purpose of this article: how to get more done with less typing, using an undocumented and underrated tool in the Clarion toolset.

Supercharged Legacy

Members of the "old school" no doubt have fond memories of the fantastic keystroke recorder in Clarion 2.1. It recorded keystrokes as you typed them, and allowed you to invoke the recorded sequence at the press of a hotkey. The only limitation was you couldn’t record more than 40 keystrokes at once. But you could save the recordings for recall in later CPD sessions.

Many sorely miss this feature in the Clarion for Windows environment. But is it really missing? Or is it just unfinished and undocumented? I stumbled across this amazingly simple yet most useful utility by accident about a year ago. Pressing Ctrl + = while in the Clarion editor causes a window to popup that asks for the keystroke needed to re-invoke the session that is about to be recorded. Click "OK" and the window disappears and recording begins. To stop recording, press Ctrl + = again.

Unfortunately, at this time the macro recording is lost when the Clarion5 IDE is closed. There was at one stage a set of replacement Clarion5 DLLs that saved the macros, but the ability to save these recordings is still not built into the Clarion IDE. With this in mind, the obvious question is, what can this be used for? And what can this thing do that the search and replace engine can't? Lets look at two examples to find out.

Example 1: Using The Macro Recorder On Its Own

Lets say you have a whole lot of fields to process in a file, and they all need to be addressed in a similar way, using vanilla Clarion and no reference statements. This example is a completely senseless piece of programming logic, but the emphasis is on using the editor and not elegant programming style!

Each field will need to be evaluated and depending on the contents a certain action has to be taken. Logic:

> *If the value = 1, clip the value. If it equals 2 then clip it and append to it the next field, with the resultant value assigned to yet another field.*

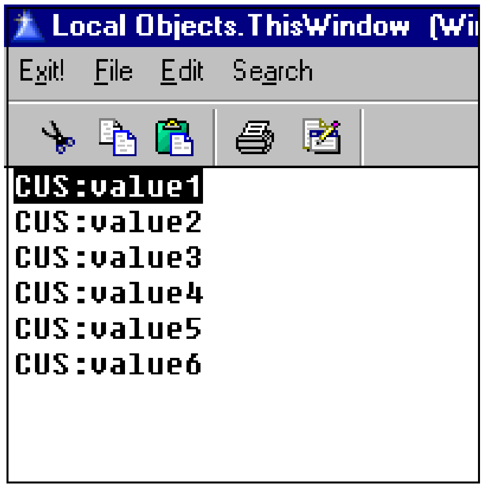
Obviously a conditional structure is needed for each field. The following case statement will do:

```
Case Cus:Value1
   Of 1
      Cus:Value1 = Clip(Cus:Value1)
   Of  2
      Cus:OtherField = Clip(Cus:Value1) & Cus:Value2
End !Case
```

Imagine if there were 20 fields like this? A programmer might need to set aside an hour or so for writing and testing. It could well be that the code gets typed out once, and gets pasted 20 times and the case statement gets manually changed 20 times. Or...use the macro editor!

The first thing to do here is to make the fields available in the editor itself. Use the "Populate Field" toolbox to dump all fields at the top left of the editor page (say, in an embed point) and highlight the first field. Delete any "white space" after the last field.

Figure 1. Editor page with fields to process and the first field highlighted.

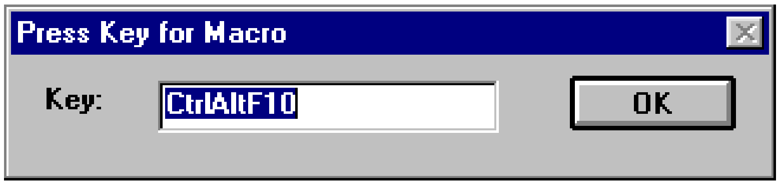


```
CUS:value1
CUS:value2
CUS:value3
CUS:value4
CUS:value5
CUS:value6
```

Turn Caps Lock off and if necessary press Insert to toggle the editor into Insert mode (as opposed to Overwrite mode). Place the cursor at line 1 character 1 and highlight until the end of the line, as shown in figure 1.

Now type **Ctrl+=** to open the macro recorder, select a hotkey (lets say **CtrlAltF10**) and click OK to start recording:

Figure 2. Macro Recorder asking for the invocation keystroke.



The following is a listing of exact keystrokes to press while recording the keystrokes. Square brackets [] indicate a single keystroke, for example:

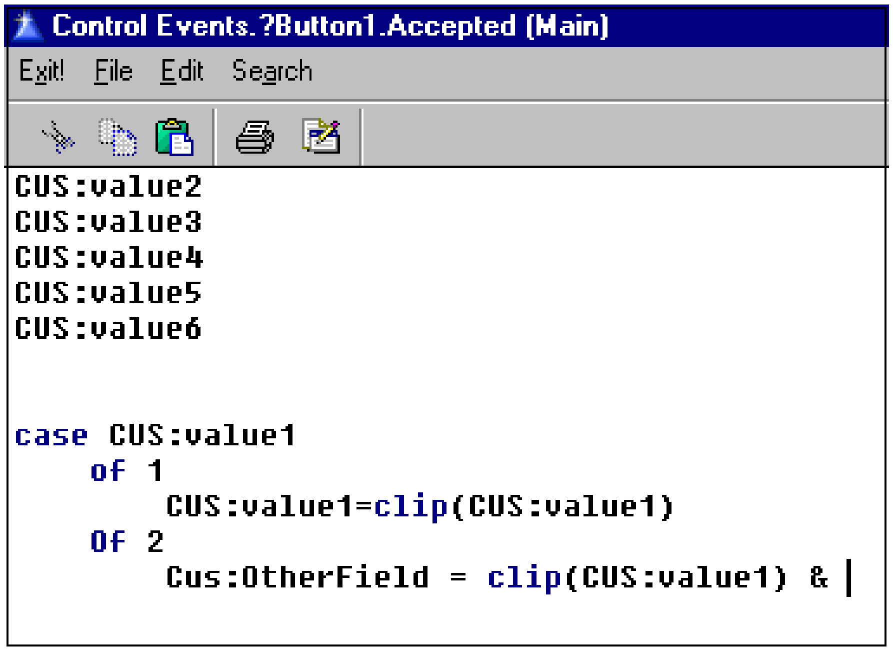
Text to type	**Description**
[Ctrl+C][Tab]	copy field name to clipboard and move to next tab

This indicates that you should hold down the Control button and press the C button and then release both, and then press the Tab button. Do not type in the brackets or the comment! Anything not in square brackets should be typed literally.

Text to type	**Description**
[Ctrl+C]	copy field name to clipboard
Ctrl+End	Go to bottom of text
[Enter(2)]	Give 2 lines of white space
Case[space][Ctrl+V]	First line of Case statement
[Enter][Tab]	New line and indent
Of 1	First option in case statement
[Enter][Tab]	New line and indent
[Ctrl+V]=clip([Ctrl+V])	clip the current field that is currently on clipboard
[Enter][Home][Tab]	New line and indent
Of 2	Second option in case statement
[Enter][Tab]	New line and indent
Cus:OtherField = Clip([Ctrl+V]) &	

At this stage the editor should look like Figure 3.

Figure 3. Partially completed case statement.



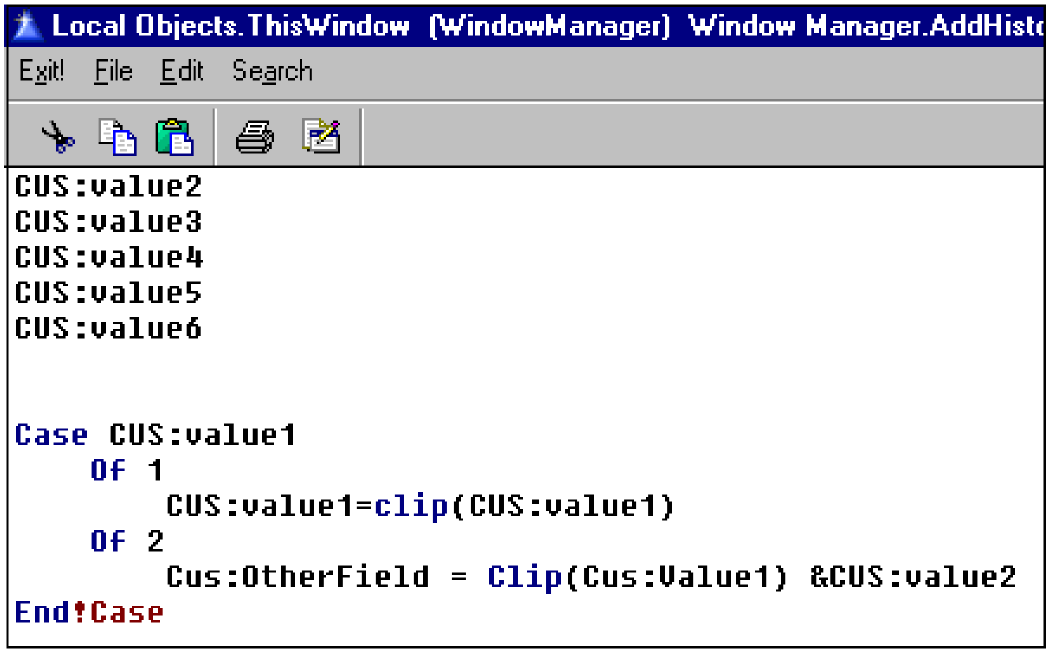
```
CUS:value2
CUS:value3
CUS:value4
CUS:value5
CUS:value6


case CUS:value1
    of 1
        CUS:value1=clip(CUS:value1)
    Of 2
        Cus:OtherField = clip(CUS:value1) &
```

Now here is a problem - the next field to type is one that *will change every time this macro is used!* The solution is simple: go and get it at the top of the editor box!

Text to type	**Description**
[Ctrl+Home]	Go to Top left hand side of editor box
[Shift+End]	Highlight the field name
[Ctrl+C]	Copy the field name to the clipboard
[Ctrl+End]	Go back to the end of the document again
[Space]	Put in a space
[Ctrl+V]	Paste the clipboard field
[Enter+Home]	New line starting at character one
End!case	Terminate case statement

Your case statement should now look like Figure 4.

Figure 4. Completed case statement for the first field.


Local Objects.ThisWindow (WindowManager) Window Manager.AddHist
Exit! File Edit Search

```
CUS:value2
CUS:value3
CUS:value4
CUS:value5
CUS:value6


Case CUS:value1
    Of 1
        CUS:value1=clip(CUS:value1)
    Of 2
        Cus:OtherField = Clip(Cus:Value1) &CUS:value2
End!Case
```

The recording is not finished though. Highlight the first field in the text editor to reset the editor into a similar state under which the recording started. This final step is critical to ensure that the macro can be repeatedly called without user intervention .

Text to type	**Description**
[Ctrl+Home]	Go to Top left hand side of editor box
[Shift+End]	highlight the field name

Press **Ctrl + =** to finish recording.

If you press CtrlAltF10 again, you will find that Cus:Value2 is also processed - and perfectly.

This will work only until the last field, because during the last macro "cycle", when the macro goes to the top of the page to copy the subsequent field there's nothing there, so the field previously copied to the clipboard is used.

Example 2: Synergy!
Using The Clipboard And Search Engine

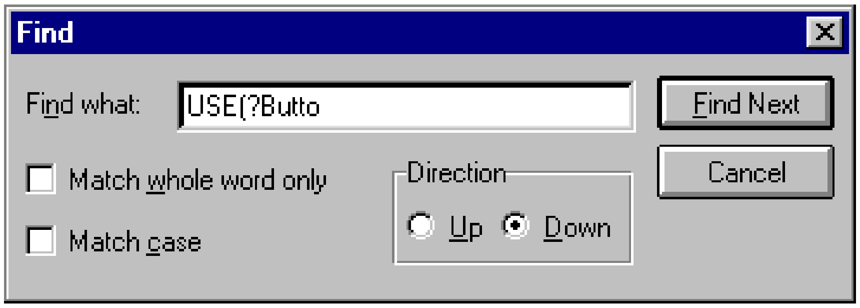
It was simple enough using the macro recorder in Example 1, because the field to be copied to the clipboard was easily located at the top of the page. Sometimes something needs to be searched for and then processed. Consider Figure 6. The task here will be to make all the buttons in the toolbar flat. The standard way is to click on each of the 20 buttons in the Screen Designer and set the "Flat" checkbox to true. More adventurous programmers might go into the window definition source code and insert the code manually. That is no problem, just slow. Using the macro engine, it takes 20 seconds to record the macro and 3 seconds to apply it to the screen.

One of the many ways to make a macro for this task would be to search for a keyphrase in each line and on finding it reacting accordingly. One such keyphrase is the following:

```
USE(?Butto
```

Searching for this text each time will without fail place the cursor on a certain line at a predictable place – and it is this predictability in the editing process that makes the use of a macro editor feasible. So, before starting a new recording, prime the search function with the phrase, and search for the first occurrence of it and press escape. (see Figure 5)

Figure 5. Priming the Clarion editor's Search Function.

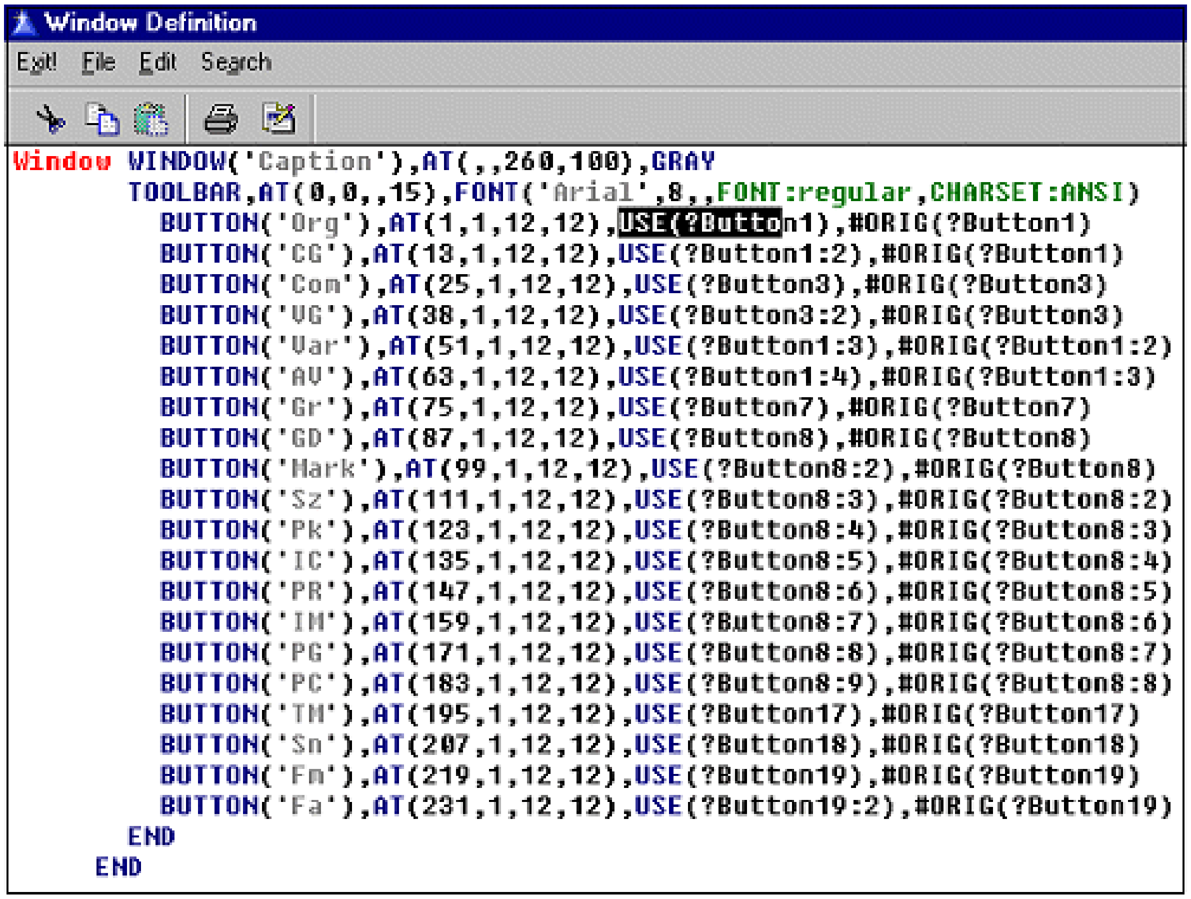


```
USE(?Butto
```

Now, the phrase is locked in the editor’s search "memory" and pressing F3 will invoke a subsequent search of the phrase to be highlighted in the editor, as displayed in figure 6.

Text to type	**Description**
[Ctrl+Home]	Go to Top left hand side of editor box
[F3]	Find next occurrence and highlight it

Figure 6. The editor's Search Function highlighting search results after pressing F3.



```
Window WINDOW('Caption'),AT(,,260,100),GRAY
         TOOLBAR,AT(0,0,,15),FONT('Arial',8,,FONT:regular,CHARSET:ANSI)
           BUTTON('Org'),AT(1,1,12,12),USE(?Button1),#ORIG(?Button1)
           BUTTON('CG'),AT(13,1,12,12),USE(?Button1:2),#ORIG(?Button1)
           BUTTON('Com'),AT(25,1,12,12),USE(?Button3),#ORIG(?Button3)
           BUTTON('VG'),AT(38,1,12,12),USE(?Button3:2),#ORIG(?Button3)
           BUTTON('Var'),AT(51,1,12,12),USE(?Button1:3),#ORIG(?Button1:2)
           BUTTON('AV'),AT(63,1,12,12),USE(?Button1:4),#ORIG(?Button1:3)
           BUTTON('Gr'),AT(75,1,12,12),USE(?Button7),#ORIG(?Button7)
           BUTTON('GD'),AT(87,1,12,12),USE(?Button8),#ORIG(?Button8)
           BUTTON('Mark'),AT(99,1,12,12),USE(?Button8:2),#ORIG(?Button8)
           BUTTON('Sz'),AT(111,1,12,12),USE(?Button8:3),#ORIG(?Button8:2)
           BUTTON('Pk'),AT(123,1,12,12),USE(?Button8:4),#ORIG(?Button8:3)
           BUTTON('IC'),AT(135,1,12,12),USE(?Button8:5),#ORIG(?Button8:4)
           BUTTON('PR'),AT(147,1,12,12),USE(?Button8:6),#ORIG(?Button8:5)
           BUTTON('IM'),AT(159,1,12,12),USE(?Button8:7),#ORIG(?Button8:6)
           BUTTON('PG'),AT(171,1,12,12),USE(?Button8:8),#ORIG(?Button8:7)
           BUTTON('PC'),AT(183,1,12,12),USE(?Button8:9),#ORIG(?Button8:8)
           BUTTON('TM'),AT(195,1,12,12),USE(?Button17),#ORIG(?Button17)
           BUTTON('Sn'),AT(207,1,12,12),USE(?Button18),#ORIG(?Button18)
           BUTTON('Fm'),AT(219,1,12,12),USE(?Button19),#ORIG(?Button19)
           BUTTON('Fa'),AT(231,1,12,12),USE(?Button19:2),#ORIG(?Button19)
         END
       END
```

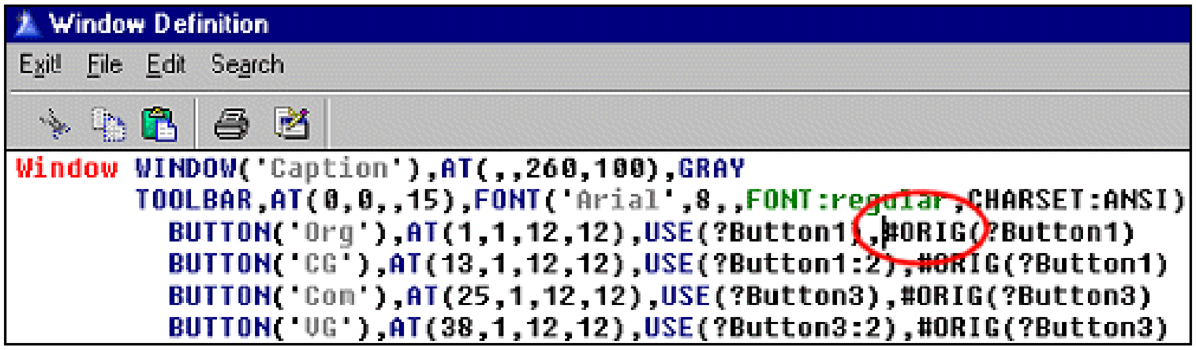
Now type **Ctrl+=** to open the macro recorder, select a hotkey (say **CtrlAltF10**) and click

OK to start recording. The search is started with the search results highlighted, to allow the programmer quick evaluation as to whether to invoke the macro each time:

Text to type	Description
Ctrl+Right]	Go to next word

Figure 7 shows the location of the cursor at a predictable position. It is at this point that the programmer will now enter the new attribute.

Figure 7. The cursor at the first attribute position of the button on the current line.



```
Window WINDOW('Caption'),AT(,,260,100),GRAY
         TOOLBAR,AT(0,0,,15),FONT('Arial',8,,FONT:regular,CHARSET:ANSI)
           BUTTON('Org'),AT(1,1,12,12),USE(?Button1),#ORIG(?Button1)
           BUTTON('CG'),AT(13,1,12,12),USE(?Button1:2),#ORIG(?Button1)
           BUTTON('Com'),AT(25,1,12,12),USE(?Button3),#ORIG(?Button3)
           BUTTON('VG'),AT(38,1,12,12),USE(?Button3:2),#ORIG(?Button3)
```

Now type the text.

Text to type	Description
Flat,	Enter the attribute
[F3]	Find next occurrence and highlight it

Press **Ctrl + =** to finish recording.

To invoke the macro, first move the cursor to the top left hand corner of the editor window and press F3 to highlight the first instance of the `USE(?Butto` text. Press **CtrlAltF10** to convert the line. After line is modified the macro will automatically highlight the next occurrence, and all you need to do is to keep pressing **CtrlAltF10** to do the next one, and so on. Click here for a sample window structure you can use to try this out.

Conclusion

There are many other uses of this utility, which you will discover, as you become more fluent with it. I recently converted a 130-table dictionary as a TXD file from using the Topspeed driver to MS-SQL by changing the driver attribute, file name and owner attribute. It worked fine (well, almost) and the whole dictionary was converted in under half an hour.

> **NOTE:** A word of caution when modifying a dictionary: Something went wrong and although the TXD imported without error into a new dictionary, and Clarion is able to use it, the Wizatron engine turns turtle when processing it, complaining of a corrupt dictionary! So before discarding a re-processed dictionary, a good idea might be to test it by running it through the Wizatron first!

The most obvious argument that people have here is "Why should I go through all that trouble for a disposable macro?" Well, ask yourself what keystrokes have been typed here that will not be typed in a non-recording scenario? Just the copying, pasting and searching is new. Everything else that was typed was typed as part of code that was needed anyway. And who knows – maybe the ability to save these macros will become part of the IDE in future versions of Clarion…

The other objection is the time taken to design these things - designing a complex macro needs a bit of gray-matter exercise! You have to *think* about it before diving in and cutting code! Sound familiar? Of course, that's what programming is about. Once you understand and practice this paradigm then macros become easy, and your coding performance skyrockets.

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

First Impression: Foundations of Clarion 5 Interactive Self-Study CD

by Patrick K. O'Brien

First Impression: Foundations of Clarion 5 Interactive Self-Study CD
(Mar 23,2000)

Adding A Class To Your ABC Program
(Mar 23,2000)

The Clarion Advisor: Adding/Removing Key Tabs
(Mar 23,2000)

The Clarion Challenge: Shoot Yourself In The Foot
(Mar 23,2000)

Because I like to learn at my own pace, and drink plenty of coffee, I've ground my share of beans while going through self-study material over the years. Unfortunately, even with caffeine, most courses put me to sleep. So I was prepared to be disappointed when I first spotted TopSpeed's new training CD at DevCon '99.

I sat down at the demo machine, without quite giving my full attention to the cheerful TopSpeed employee who was explaining things to me. My goal was to click on a few of the lessons, confirm my belief that this was a waste of my time, say "Thank you" to the cheerful employee, and be on my way. Instead, I walked away with a training CD in my conference bag, and a few less dollars in my wallet.

Overview

TopSpeed's Interactive Self-Study products follow the philosophy that people learn best through a combination of reading, watching and doing. To that end the Foundations CD comes with material to read, videos to watch (and hear), and exercises to do. The real strength of the package comes from the fast-paced, narrated video clips. (The CD contains 81 AVI files totaling 260 MB.).

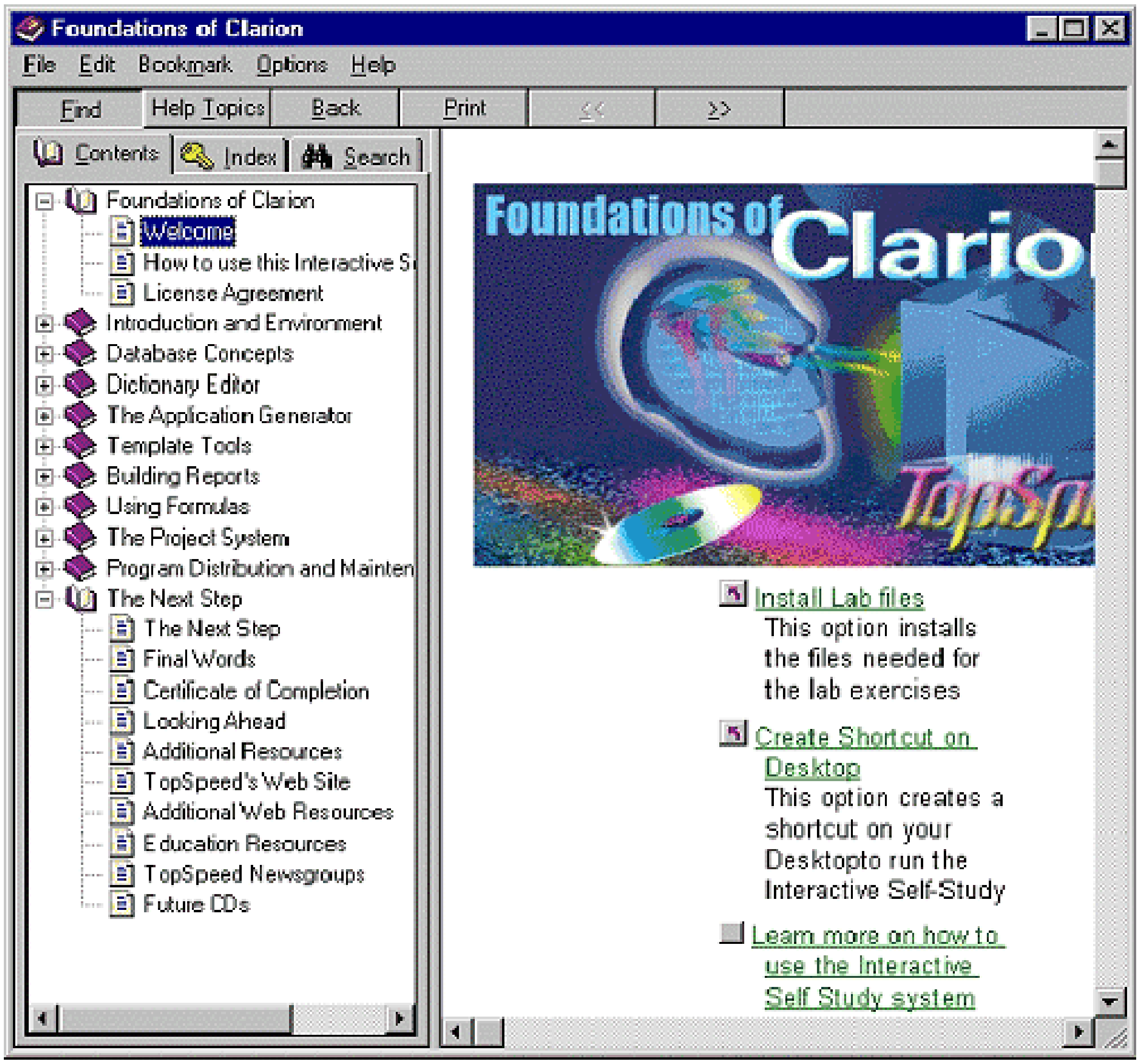
I can best describe the experience as a whirlwind tour of Clarion, lead by a smooth-talking tour guide, Bob "Click on the link below to continue the lesson" Foreman (Bob is a TopSpeed employee and the voice on the videos). In any good tour, the guide should do more work than you, and you should walk away having seen and learned more than you would have by stumbling around on your own. I think the Foundations CD lives up to this standard.

Installation

Installation is simple. In fact, it isn't even necessary. You can just pop the CD into your computer and start going through the lessons. That's because the lessons are delivered in the form of a standard Windows Help file (WinHelp 2000, to be precise). This also means that the entire course has the same Contents, Index, Search and Bookmark features with which most Windows users are already familiar. (See Figure 1)

You have the option of installing files that go along with the lab exercises. Each exercise has *lab* folders that provide a starting point for the exercises in the workbook without having to start from scratch, *solution* folders showing how the final projects should have turned out (including EXE files so you can even see where you are heading before you start the exercise), and possibly some *example* folders containing additional variations discussed in that exercise.

Figure 1. Foundations of Clarion CD.



Foundations of Clarion
- Welcome
- How to use this Interactive S
- License Agreement

Introduction and Environment
Database Concepts
Dictionary Editor
The Application Generator
Template Tools
Building Reports
Using Formulas
The Project System
Program Distribution and Mainten
The Next Step
- The Next Step
- Final Words
- Certificate of Completion
- Looking Ahead
- Additional Resources
- TopSpeed's Web Site
- Additional Web Resources
- Education Resources
- TopSpeed Newsgroups
- Future CDs

Install Lab files
This option installs the files needed for the lab exercises

Create Shortcut on Desktop
This option creates a shortcut on your Desktopto run the Interactive Self-Study

Learn more on how to use the Interactive Self Study system

Lessons

The Foundations CD contains 10 lessons:

- Course Introduction and Environment Overview
- Database Design
- The Dictionary Editor
- The Application Generator

- Template Tools
- Creating Reports
- Formulas and Range Limits
- The Project System
- Program Distribution and Maintenance
- Course Wrap Up and the Next Step

Each lesson contains a mixture of reading material and video clips, and ends with a self-test to make sure you were paying attention. Every so often you are encouraged to complete a lab exercise before continuing on to the next lesson.

Lab Exercises

The Foundations CD package comes with a 160 page workbook containing four lab exercises, similar in nature to the tutorials found in the standard TopSpeed documentation:

- The Data Dictionary
- Creating An Application
- Enhancing The Application
- Reports

The lab exercises actually lead you through the development of an entire application, a Windows Contact Manager. So even though the lab exercises are broken up into four pieces, they are really more like four phases of a larger project. And each exercise is integrated into the material presented in the training CD, such that you learn a little, apply a little, learn some more, apply some more, until, by the end, you’ve created a complete, working (albeit relatively trivial) application.

Studying In Your Pajamas

Exposure is the key concept here. Exposure to Clarion, that is, which is what this CD does best. It would take you much longer to become familiar with all facets of the Clarion IDE on your own.

You can sit back, drink your coffee, and relax while Bob Foreman and company build applications right before your eyes. "To continue the lesson, click on the link below." I actually ignored that advice throughout the lessons. I suffer from too much mouse-clicking as it is. Instead, I used the > (greater than) key to move to the next lesson, and the page down key to scroll through each lesson. Because the lessons are set up to be followed in a linear fashion, you can go through the entire course with a minimum of effort. When I felt like stopping, I would create a bookmark to hold my place, allowing me to pick up right where I left off.

As I went through the lessons, I realized how much I had already learned about Clarion on my own. This makes it a little difficult to give a completely objective review of this product. I can’t really fault the product for covering familiar territory; I only wish it had been available when I first started using Clarion earlier this year. I truly believe that it would have saved me a great deal of time and effort. If you are brand new to Clarion, or have never really worked with all that the Clarion IDE has to offer, I think you will find the Foundations CD to be both professional and enjoyable, and certainly less expensive than a trip out of town to attend a training class.

Licensing

First of all, I am neither a lawyer nor a representative of TopSpeed. But if you are like me, you want to know what the rules are when it comes to using this product. To help answer that, I will simply quote a couple of relevant passages from the license agreement:

- "You may install and use one copy of the SOFTWARE PRODUCT, or any prior version for the same operating system, on a single computer."
- "Rental. You may not rent, lease, or lend the SOFTWARE PRODUCT."
- "Software Transfer. You may permanently transfer all of your rights under this LICENSE, provided you retain no copies, you transfer all of the SOFTWARE PRODUCT (including all component parts, the media and printed materials, and this LICENSE), and the recipient agrees to the terms of this LICENSE."

Summary

The Foundations of Clarion CD is definitely targeted at new users of Clarion. Like a guided tour through a new city, it does an excellent job demonstrating the many facets of the Clarion toolset. Experienced users won't find enough to sustain their interests, but new users should find the CD enjoyable, and appreciate the shortcut this CD provides in their quest to summit the learning curve we've all had to climb. The lessons never take on that condescending tone common in other training products. The Foundations CD is professional and enjoyable, with a nice, snappy pace.

Foundations of Clarion 5 Interactive Self-Study CD lists for US$495.00

The Foundations CD is available directly from TopSpeed Sales. Additional information, system requirements and instructions for ordering can be found at the TopSpeed website.

Patrick O'Brien is a partner with Orbtech, where he occasionally dabbles in software development using Wizatrons and Internet technology.

Clarion MAGAZINE

Adding A Class To Your ABC Program

By Tom Ruby

First Impression: Foundations of Clarion 5 Interactive Self-Study CD
(Mar 23,2000)

Adding A Class To Your ABC Program
(Mar 23,2000)

The Clarion Advisor: Adding/Removing Key Tabs
(Mar 23,2000)

The Clarion Challenge: Shoot Yourself In The Foot
(Mar 23,2000)

I’ve seen several articles in Clarion Magazine on how or why we should all write classes, but somehow I was still left wondering how to use such a class in my ABC programs. After a painful week on the newsgroups trying to sort this out (thanks Arnor, Dave, Jeff, Paul, and Alexey), I thought I’d put the "how to" in a single article where it would be easy to find. In addition, you get a working object for dealing with COM and LPT ports in 32 bit programs. What a deal!

In an ABC program, you make a new object with two files: an .INC file which contains the class definition, and a .CLW file which contains the source of the methods. (Remember that an object is just the name for an instance of a class.) You put these files in the \LIBSRC subdirectory. You can put these files in the application directory so the class only applies to some applications, but this makes life a lot more difficult, especially with multi-dll applications

It’s easiest if you make your classes conform to the ABC library specification. The tricky parts come at the top and bottom of the .INC file and the top of the .CLW file and on the class definition. Here is my basic com port object definition:

```
PortClass     CLASS,TYPE
Handle          SHORT
NameCString     CSTRING(255)
LastError       ULONG
DCB             GROUP
Length            ULONG
BaudRate          ULONG
fBinary           ULONG
fParity           ULONG
fOutxCTSFlow      ULONG
fOutxDSRFlow      ULONG
fDTRControl       ULONG
fDSRSensitivity   ULONG
```

```
fTxContinueOnXOff ULONG
fOutX             ULONG
fInX              ULONG
fErrorChar        ULONG
fNull             ULONG
fRtsControl       ULONG
fAbortOnError     ULONG
Dummy2            ULONG
wReserved         USHORT
XonLim            USHORT
XoffLim           USHORT
ByeSize           BYTE
Parity            BYTE
StopBits          BYTE
XOnChar           BYTE
XOffChar          BYTE
ErrorChar         BYTE
EvtChar           BYTE
wReserved2        USHORT
                END
Init            PROCEDURE( STRING PortName ),SHORT
SetUp           PROCEDURE(),SHORT
SetUpString     PROCEDURE( STRING PortString ),SHORT
Kill            PROCEDURE( )
ClearIncoming   PROCEDURE(),SHORT
ClearOutgoing   PROCEDURE(),SHORT
Read            PROCEDURE( *CSTRING buffer, ULONG bytes, ↵
                  *ULONG BytesRead ),SHORT
Write           PROCEDURE( *CSTRING buffer, ULONG bytes )↵
                            ,SHORT
SetTimeouts     PROCEDURE( ULONG ReadInterval, ↵
                    ULONG ReadMultiplier, ↵
                    ULONG ReadConstant, ↵
                    ULONG WriteMultiplier, ↵
                    ULONG WriteConstant ),SHORT
NormalTimeouts  PROCEDURE(),SHORT
              END
```

The idea is to call the `INIT` method with the name of the port, i.e.`SomePort.Init( ‘COM1:’)`. Then you can stuff data into the Device Control Block (DCB), set the baud rate and such with `SomePort.DCB.BaudRate = 9600`, and call `SomePort.Setup`. Since the API provides a setup string call, you can also call `SomePort.SetupString(‘baud=9600 parity=N data=8 stop=1’)` with a string read from a configuration file or other source. Then it’s a simple matter to use the Read and Write methods to send and receive data. If you want to send data directly to a printer, you can use:

SomePort.Init(‘LPT1:’)

And then write to it with `SomePort.Write`.

The INC File

To get all this into an ABC program, you need to add some things to the definition in the .INC file. Start out with:

```
!ABCIncludeFile
OMIT('_EndOfInclude_',_PortClassPresent_)
_PortClassPresent_ EQUATE(1)
```

The first line tells the IDE that this is an ABC class. The IDE looks for this when it's "Reading ABC header files."

The next two lines tells the compiler to skip the file's contents if it's already been included. It checks for an equate, and if it's not zero, it skips till it finds `'_EndOfInclude_'` at the bottom of the include file. So at the bottom of the include file, you better have a line which says `'_EndOfInclude_'`.

Next, you have to alter the class definition by adding the module and link information to the class definition:

```
PortClass CLASS,TYPE,MODULE('Ports.clw'),↵
LINK('Ports.clw',_ABCLinkMode_),DLL(_ABCDLLMode_)
```

Now you can type all this, but it's a lot better to copy and paste it from an existing .INC file because if you mistype something, like omit the last underscore, your class won't work and it will take somebody of Alexey's caliber to find it (thanks, Alexey). Notice the `MODULE` and `LINK` attributes name the .CLW file that holds the method code.

The rest of the addition deals with multiple DLL projects and sets the DLL and the LINK attributes properly. If you're building a data DLL and check "Export template globals and ABC's as external" the methods will become external methods, that is, code contained in another DLL. In your data DLL, these will automagically be exported for you.

The CLW File

The CLW file starts with a `MEMBER` statement. Don't put this in column 1. MEMBER either has a parameter after it telling the compiler which application's main source file contains needed global definitions, or it has no parameter, meaning it can be used with any application.

The `MAP` section is very important. If you don't have anything to prototype here, you'd better put an empty `MAP` and `END`, or the whole works will blow up on you. If you have DLL or Windows API methods to prototype, this is where they go.

After the map section come two equates:

```
_ABCDllMode_ EQUATE(0)
_ABCLinkMode_ EQUATE(1)
```

I don't really understand these equates as I thought the templates set these for you, but the guys at the Development Centre know a lot more about it than I do, so just do as they do.

Next, you have to include the .INC file. This gets the object prototyped in the module that contains the code.

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

Below that comes the code for the methods.

Getting It Into Your app

You have to include the class at the MODULE level. While you're looking at your app's procedure tree, highlight the procedure where you need your object and switch to the module view. You'll see the module that contains the procedure. Double click on this module, and you'll get a little window. Hit the embeds button. In the Start of Module embed, put your include statement (not in column 1). In this case,

```
INCLUDE('ports.inc')
```

will get the class defined for all the procedures in the module.

Next, you need to put one or more objects in your procedure. You can do it in the data section, which is handy because your object shows in the field box of the embed editor. Just add a variable using the DATA button, set its type to CLASS, and the base type to your class name. If you need to add any data fields to your class, you can put them in this list before the `END` statement. You can also put the object in the DATA embed for the procedure.

Summary

So there you have it, in one place: instructions for adding a class to your ABC program. There's a zip file that contains my ports class and a silly little app to demonstrate and test it.

The main points are:

- Put the object definition in an .INC file in the LIBSRC directory.
- Put the method code in a .CLW file in the LIBSRC directory.
- At the top of the .INC file, put 3 lines:
  ```
  !ABCIncludeFile
  OMIT('_EndOfInclude_',_YourClassPresent_)
  _YourClassPresent_ EQUATE(1)
  ```
- Add to your class definition(s):
  ```
  ,MODULE('YourClass.clw'),LINK('YourClass.clw',↵
  _ABCLinkMode_),DLL(_ABCDLLMode_)
  ```
- At the bottom of the .INC file, put a line like:
  ```
  _EndOfInclude_
  ```
- At the top of your .CLW file, you need to put:

  A line which says MEMBER

  A map, even if there's nothing to prototype
  ```
  _ABCDllMode_ EQUATE(0)
  _ABCLinkMode_ EQUATE(1)
  INCLUDE('Ports.INC'),ONCE !Not in the first column
  ```

That's it. Enjoy!

Download the source

Tom Ruby, who is no relation to the man who shot Lee Harvey Oswald, is an independent contractor living in the middle of a hayfield in Central Illinois with his wife Susan and two red-headed sons, Caleb and Ethan. He has been using Clarion *for Windows since the summer of '95. Before that, he was a "TopSpeeder" using Modula II, so he has never used the DOS versions of Clarion.*

The Clarion Advisor: Modifying Browse Sort Tabs

by Dave Harms

First Impression: Foundations of Clarion 5 Interactive Self-Study CD
(Mar 23,2000)

Adding A Class To Your ABC Program
(Mar 23,2000)

The Clarion Advisor: Adding/Removing Key Tabs
(Mar 23,2000)

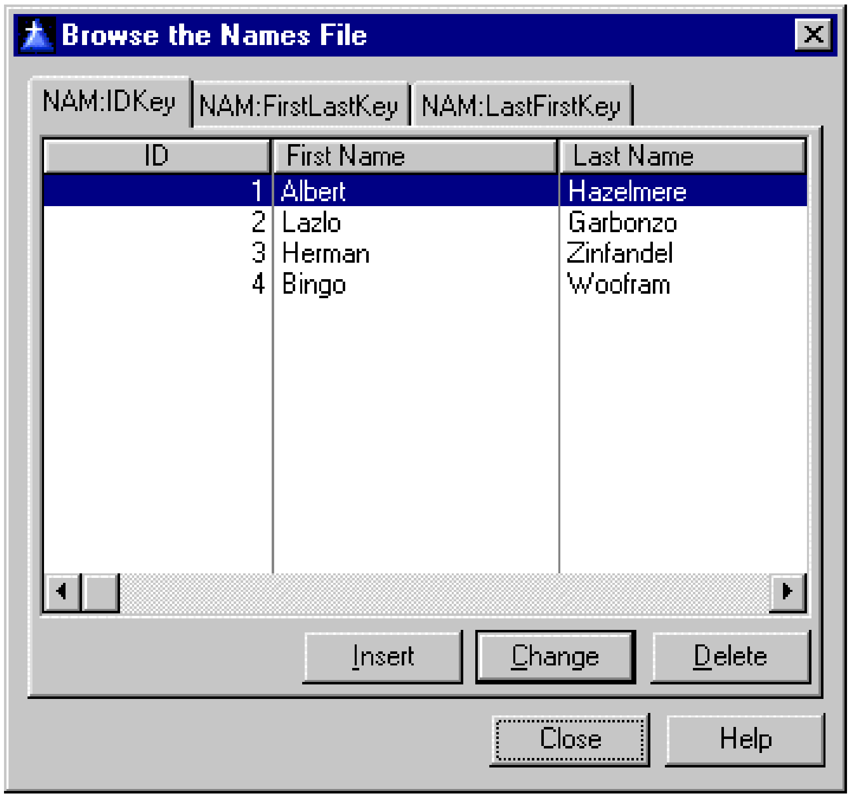
The Clarion Challenge: Shoot Yourself In The Foot
(Mar 23,2000)

Clarion's Application Generator does a great job of creating tabbed browses, with one tab for each key. But what happens when you want to change the order of the tabs on a browse, or add/remove one or more tabs? If you're not careful, you'll end up with a browse that doesn't work the way you expect.

The following approach can be applied to both ABC and Legacy apps.

Consider a simple file with three fields: an ID; a person's first name; and a person's last name. If you set up a key for each of these fields (make the ID key unique and autonumbering), the browse wizard will create a browse with three tabs, similar to that shown in Figure 1.

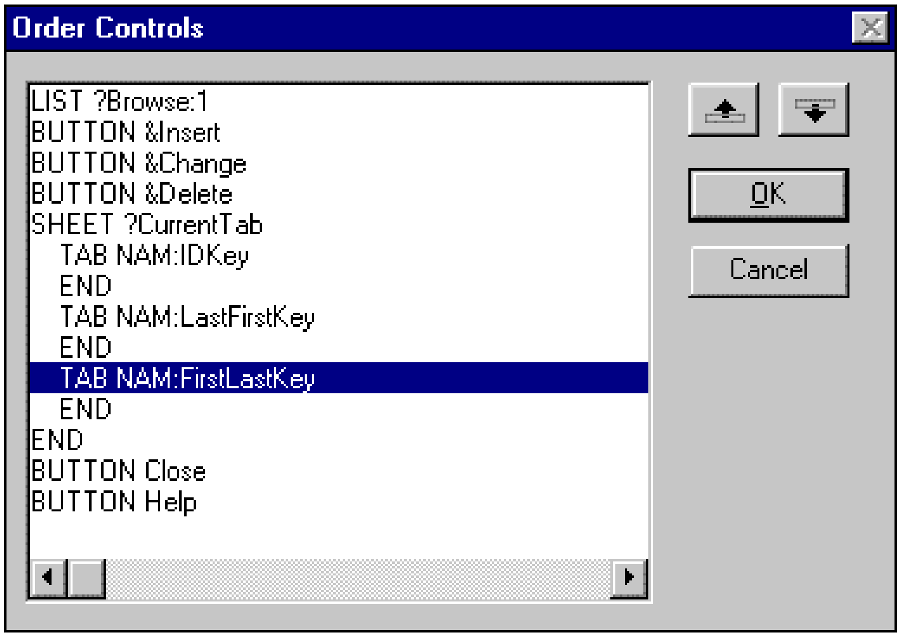
Figure 1. A three-tabbed browse.



ID	First Name	Last Name
1	Albert	Hazelmere
2	Lazlo	Garbonzo
3	Herman	Zinfandel
4	Bingo	Woofram

Let’s say you want to switch the order of the second and third tabs. In the window formatter you can choose Edit|Set Control Order and move the FirstLastKey tab down to the last position, as shown in Figure 2.

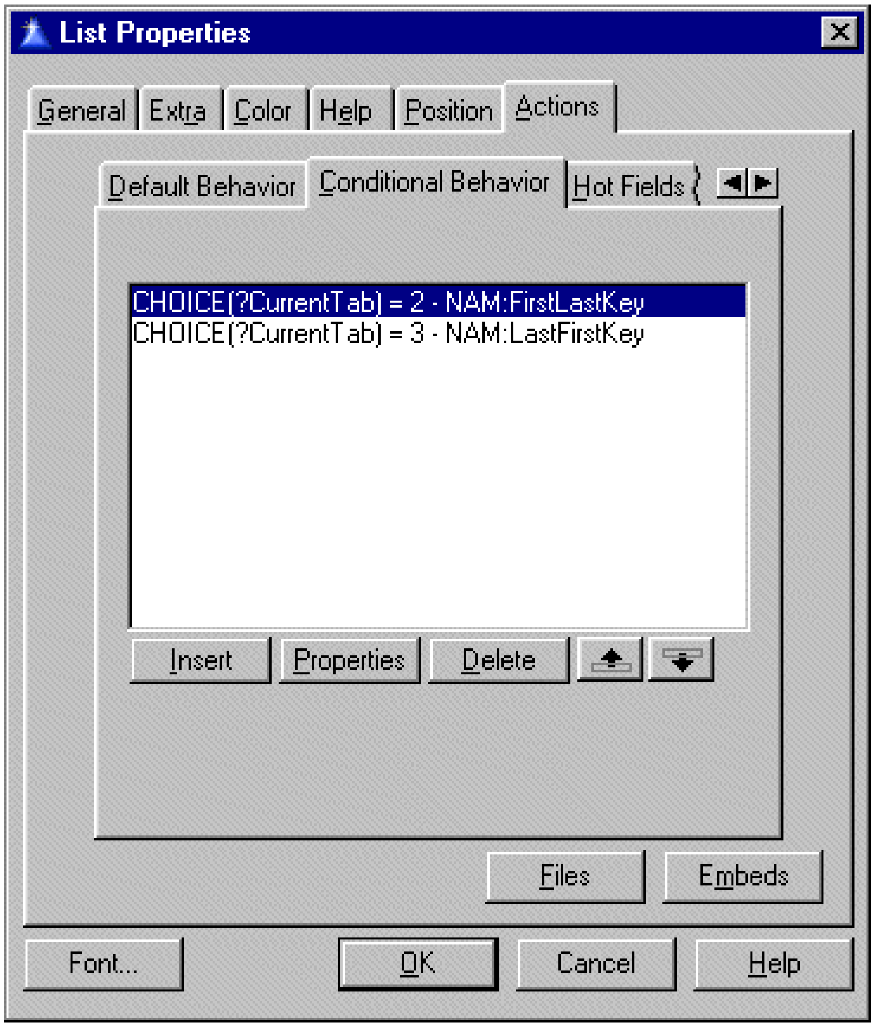
Figure 2. Changing the control order.



```
LIST ?Browse:1
BUTTON &Insert
BUTTON &Change
BUTTON &Delete
SHEET ?CurrentTab
   TAB NAM:IDKey
   END
   TAB NAM:LastFirstKey
   END
   TAB NAM:FirstLastKey
   END
END
BUTTON Close
BUTTON Help
```

The problem is that when you run the program, the text on the tabs has changed but the behavior hasn’t. To find out why, look at the Conditional Behavior tab on the browse’s Actions tab.

Figure 3. The browse Actions|Conditional Behavior tab.



```
CHOICE(?CurrentTab) = 2 - NAM:FirstLastKey
CHOICE(?CurrentTab) = 3 - NAM:LastFirstKey
```

Generated browses assign the key set in the file schematic to the first tab in the browse. All other tabs have code generated to set the sort order based on the value returned by the `CHOICE` function, which simply returns the number of the selected tab. This means that sort orders, as generated, are relative to which number tab has been selected, not the text on that tab.

Updating The Tab Order

There are several ways to address this problem. One is to simply switch the numbers in the text. Click on the properties for each conditional behavior and in the Condition field, where you see `CHOICE(?CurrentTab) = x,` change `x` to the appropriate number. You can instead change the key but if you have other settings such as a filter, locator or range limit that's a lot more work.

If you want to avoid the dependence on tab order you can change the Condition entry to something like `?CurrentTab{PROP:ChoiceFEQ} = ?Tab:3` where `?Tab:3` is the field equate for the tab - `PROP:ChoiceFEQ` uses the field equate name rather than the tab order. (You might want to change the automatically generated field equate for the tab to something more descriptive.) With `PROP:ChoiceFEQ` you can move the tabs anywhere you like without having to change your code.

Keep in mind that the first tab on the browse corresponds to the default settings for the browse, and the key used is that specified for the browse. If you want to move a different tab to the first spot you'll need to update the default settings as well as the conditional settings.

Adding And Deleting Tabs

Deleting a key tab from a browse is easy. Click on the tab (not the sheet, just the tab!) and delete. You should also delete the corresponding conditional entry from the template. Adding a sort tab simply involves adding a new tab to the sheet, and then making the appropriate entries in the browse’s Conditional Behavior tab.

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.*

INVEST in your own abilities

Clarion magazine

published by CoveComm Inc.

Search

Clarion MAGAZINE

The Clarion Challenge

First Impression: Foundations of Clarion 5 Interactive Self-Study CD
(Mar 23,2000)

Adding A Class To Your ABC Program
(Mar 23,2000)

The Clarion Advisor: Adding/Removing Key Tabs
(Mar 23,2000)

The Clarion Challenge: Shoot Yourself In The Foot
(Mar 23,2000)

From time to time Clarion Magazine holds programming contests. Inspired by a discussion in the Topspeed.Topic.General newsgroup, we've decided to do something a bit different. The following is one version of a popular description of how to use various languages to shoot yourself in the foot. Conspicuously missing is an entry for Clarion.

Compose a Clarion version, send it to editor@clarionmag.com, and become famous!

Task: Shoot yourself in the foot

C: You shoot yourself in the foot.

C++: You accidentally create a dozen instances of yourself and shoot them all in the foot. Providing emergency medical assistance is impossible since you can't tell which are bitwise copies and which are just pointing at others saying "That's me, over there."

FORTRAN: You shoot yourself in each toe, iteratively, until you run out of toes, then you read in the next foot and repeat. If you run out of bullets, you continue with the attempts to shoot yourself anyways because you have no exception handling capability.

Pascal: The compiler won't let you shoot yourself in the foot.

Ada: After correctly packing your foot, you attempt to concurrently load the gun, pull the trigger, scream, and shoot yourself in the foot. When you try, however, you discover you can't because your foot is of the wrong type.

COBOL: Using a COLT 45 HANDGUN, AIM gun at LEG.FOOT, THEN place ARM.HAND.FINGER on HANDGUN.TRIGGER and SQUEEZE. THEN return HANDGUN to HOLSTER. CHECK whether shoelace needs to be re-tied.

LISP: You shoot yourself in the appendage which holds the gun with which you shoot yourself in the appendage which holds the gun with which you shoot yourself...

FORTH: Foot in yourself shoot.

Prolog: You tell the program that you want to be shot in the foot. The program figures out how to do it, but the syntax doesn't permit it to explain it to you.

BASIC: Shoot yourself in the foot with a water pistol. On large systems, continue until entire lower body is waterlogged.

Visual Basic: You'll really only appear to have shot yourself in the foot, but you'll have had so much fun doing it that you won't care.

HyperTalk: Put the first bullet of gun into foot left of leg of you. Answer the result.

Motif: You spend days writing a UIL description of your foot, the bullet, its trajectory, and the intricate scrollwork on the ivory handles of the gun. When you finally get around to pulling the trigger, the gun jams.

APL: You shoot yourself in the foot, then spend all day figuring out how to do it in fewer characters.

370 JCL: You send your foot down to MIS and include a 400 page document explaining exactly how you want it to be shot. Three years later, your foot comes back deep-fried.

Paradox: Not only can you shoot yourself in the foot, your users can too!

Access: You try and point the gun at your foot, but it shoots holes in all of your distribution disks instead.

Revelation: You're sure you're going to be able to shoot yourself in the foot, just as soon as you figure out what all these nifty little bullet-thingies are for.

Assembler: You try to shoot yourself in the foot, only to discover you must first invent the gun, the bullet, the trigger, and your foot.

Modula-2: After realizing that you can't actually accomplish anything in this language, you shoot yourself in the head.

Clarion: Yeah, why is it missing? Compose your Clarion version of the shoot-self-in-foot task and send it to editor@clarionmag.com. I'll post the best entries in an upcoming issue.

published by CoveComm Inc.

Clarion MAGAZINE

Tool Talk: Reviews News

Tool Talk: Reviews News
(Mar 28,2000)

Editorial
(Mar 28,2000)

The Clarion Advisor: Browse Popup Menu Tricks
(Mar 28,2000)

Subclassing With A Twist
(Mar 28,2000)

March 2000 News
(Mar 28,2000)

By Tom Hebenstreit, Reviews Editor

Just this last week, I received an email from TopSpeed announcing a new "Application Review" service. In case you didn't get a copy of it yourself, here is the gist of it:

> *Introducing the TopSpeed Application Review. Send in your application to TopSpeed and we will perform a thorough review of your application. Together we will look over the application and provide a written critique on areas where we feel your application could be improved upon as well as tips for possible improvements.*
>
> *If your application then meets all of TopSpeed's standards, we will provide a Certificate stating precisely that in addition to the report on your application.*

Naturally, this brought up more than a few questions in my mind, and being the inquisitive type, I immediately fired off a return email asking for more details.

By the next day, I had not only a nice response to my questions (courtesy of Christina Downs, TopSpeed Consulting Sales Manager), but also a Word document that laid out the Application Review process in much greater detail.

Here are my questions, along with Christina's responses (in italics):

> What is the basic pricing structure, i.e., do you charge by the procedure, lines of code or what?
>
> *The cost is $500 for one application, for multiple applications would be $100 per hour with a minimum of 5 hours.*
>
> Do you provide this service for legacy template applications?
>
> *Yes.*
>
> Do you sign NDA's (Non-Disclosure Agreements) and non-competes so that the developers who send you materials can protect their properties?
>
> *Yes.*

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

How long (on average) would the process take?

Minimum of 5 hours, but not limited to.

Who actually looks at the code? Just wondering how useful it would be to have the code looked at by someone with less experience than *I* have.<g>

TopSpeed Consultant

It's nice to see that they don't limit the service to only ABC based applications, and that they are willing to sign an NDA. The experience question isn't really answered, so I guess one just has to trust that a TopSpeed Consultant will be well trained.

Let's look a bit closer at the process, starting with what you would have to provide to TopSpeed.

You give them:

- A working copy of the application itself.
- A description of the application and intended usage, e.g. networked or not, single or multi-user, etc.
- All supporting data files needed for correct operation (and so that they can do some general performance testing).
- All app and dictionary files needed to compile the application.
- All third-party tools used (once again, so that they can compile the app).
- Instructions on how to use the application, along with any special instructions regarding areas where you would like special in-depth evaluation.

Regarding that first point about a working application - this may seem a bit obvious, but it should be stressed that the Application Review service is *not* for helping you debug your applications. It is an independent review of the current state of your application. Sending them a broken app would simply result in them giving your app a poor review. (I'm sure that TopSpeed Consulting would be delighted to take your money in return for helping with actual programming services, but that's another topic…)

The process might also get a bit tricky if you tend to use a lot of third-party tools. I'm not sure how they plan to deal with the issues of installing those products (I assume you would have to provide your personal installs, unlocks, passwords, serial numbers, etc.). It could take a fair amount of time to actually set up a Clarion system so that all of the options are available and functional.

It would also appear that if the TS Consultant has no experience with a particular set of third party tools, the most they could do is just take it on faith that you have used those templates correctly. There are a *lot* of tools out there, folks.

So, what do you really get for your money? Here is the summary from the TopSpeed document I was sent:

> *TopSpeed will perform a 5 hour evaluation of a customer application. This evaluation will concentrate on adherence to Clarion programming guidelines rather than customer design specifications. While the review will check for Microsoft standards, TopSpeed will not make a judgment about the effectiveness of an applications design. The review will concentrate on three particular areas: Database Design, Program Maintainability, and Program Functionality.*

In brief, here are the high points of each area of analysis:

Database Design deals with issues such as proper use of data types, normalization, keys, referential integrity, naming conventions, file driver selection and other dictionary based concerns.

Program Maintainability includes issues such as clarity of embedded code, template usage, program organization, error checking and so forth.

Program Functionality will cover items such as window design, user help, navigation, multi-user considerations and error handling among other topics.

Bottom line: Is this a good thing?

Obviously, until someone actually goes through the process, there's no way to know for sure. The concept is certainly sound, and I am a firm believer in the value of getting unbiased, outside evaluations of the code one writes. It is easy to fall into various programming traps over time, such as using implicit variables, cryptic variable names, not commenting code, not taking advantage of newer language and template features, and just generally doing things out of habit rather than as part of a considered design. Rest assured, your application will look rather different when viewed through someone else's eyes.

It might also be useful if the review pointed out areas where you could strengthen your overall design methodology (such as taking greater advantage of particular data dictionary or template features).

Taking a broader view, this type of service could be especially valuable if you are trying to bring a new commercial product to market. In that situation, I'd say it could be money well spent (assuming that the review is in fact comprehensive and useful). If all you write is in-house applications for your particular business needs, it might still be useful as an outside "reality check."

Of course, if you are cynical you might wonder if TopSpeed's ulterior motive is to try and sell you consulting. Is the App Review in-disguise marketing for TS consulting, or is it really an objective third party review? (Hmmm...I must be a cynic just for thinking such a thing!) Only time will tell, I guess.

So, would you submit an application? Yes? No? Why not? Drop me a line and give me your thoughts. I'd be especially interested in hearing from anyone who actually does go through the process.

If you would like more information, you can contact Christina Downs at: cdowns@topspeed.com

Speaking of Reviews…

We have a number of new products lined up for review here at Clarion Magazine, so I'd though I'd take a minute and let you know what's coming up.

Class Word

This is a new product from Juan Domingo Herrera and SoftMasterS. According to the preliminary information, it is a set of classes which give you the ability to interface with and control Microsoft Word from within your Clarion applications. You can create as many Word objects as you need, and manipulate them independently. Document creation, content manipulation, search and replace, creating tables, spell check and more are all available.

Class Word is also template neutral - it can be used with both ABC and Legacy templates.

LSZip 2.5

This recent update to Linder Software's LSZip package brings a number of new features to an already powerful product, including improved ABC classes, in-memory compression, new documentation and more.

In case you aren't familiar with LSZip, it allows you to create and manipulate archives that are completely compatible with the popular ZIP format.

LSZip is compatible with most versions of Clarion, including the new 5.5 betas.

UltraTree Platinum

This update to longtime standout UltraTree replaces the previous Professional Edition. New features, new support options, but the same design goal - giving you the best Tree tool around.

If you use anything more than the most rudimentary features of List Box Trees, you will be interested in this review.

UltraTree Platinum is an ABC template set.

Timesavers Scheduler Templates

This is a newer product from POSitive Software, makers of the TimeSavers series of templates. Scheduler includes two primary templates: a Calendar Creator Template, that lets you dynamically create any kind of Calendar you want; and a Scheduling Grid Template, that lets you create Monthly, Weekly, or Daily Scheduling Grids.

The Scheduler Templates are compatible with C4 and C5, ABC and Legacy.

And From CapeSoft…

These guys just don't stop (thankfully!). There are currently four CapeSoft products in the review pipeline:

Secret Agent: Allows you to add Microsoft's Agent technology into your own applications, meaning you can create your own version of the dancing paperclip so common in Microsoft's popular Office products.

Tear Off: This little utility lets you create "tear off" menus. These are menus that can be pulled off the menu bar and used as a floating, dockable toolbar.

MakeOver 2.0: An update to CapeSoft's nifty user interface tool, you need to see it to believe it. MakeOver allows you to dynamically change the entire look of an applications interface - without recompiling. Templates are also provided to allow the user to choose from various styles, similar to commercial products such as Quicken, etc.

SecWin 3.0: Lots of new features and improvements in one of the premier application security and protection template products. SecWin is currently in late beta, so it will probably come a bit later than the other reviews.

Sounds Good… So When Do I See Them?

Soon… and that means I'd better get back to work!

By the way, what would *you* like to see reviewed? I am always looking for feedback and suggestions for the Clarion Magazine review process, and would like to encourage all of our readers to write me and let me know how we can best serve your interests. Thanks!

INVEST in your own abilities Clarion magazine

published by CoveComm Inc.

Search

Clarion MAGAZINE

Editorial

Tool Talk: Reviews News
(Mar 28,2000)

Editorial
(Mar 28,2000)

The Clarion Advisor: Browse Popup Menu Tricks
(Mar 28,2000)

Subclassing With A Twist
(Mar 28,2000)

March 2000 News
(Mar 28,2000)

Clarion 5.5, with its shift from Java to Javascript on the client side, represents a major step forward for Clarion web development. Although there may yet be a day when a Java client is a viable option, present market conditions dictate lighter weight web applications, which Clarion 5.5 can deliver. These apps leverage Clarion's long history of application generation and allow developers to create complex web database applications with a minimum of effort.

Unfortunately, little is yet known about Clarion 5.5's scalability. How many continuous users can a single server handle? What are the typical memory and processor requirements? What features exist to manage multi-server configurations? What kinds of load testing has TopSpeed performed?

There has been a lot of discussion on these topics in the TopSpeed newsgroups, generally met by deafening silence from TopSpeed. Steve Parker, whose knowledge base is probably the most-used public Clarion web app, has seen 80-100 simultaneous users on a dual Pentium system, and he estimates the system could handle 200 users without difficulty (although this is a small application, the search function is very CPU-intensive). Other developers have offered sites for test purposes, or have made plans for their own internal load testing.

As helpful as all of these efforts are, they are no substitute for TopSpeed conducting or obtaining a proper benchmark of Web Builder's scalability, reliability, and resource utilization.

Web Builder needs to be stress-tested to ensure stability. The app broker reportedly reduces each application's footprint by sharing DLLs in memory (this is why NT reports more memory in use than actually is in use). If one application tramples shared memory, other applications can be affected. How does the broker recover from this kind of error, if at all? How fault tolerant is the broker when it encounters a simple application error?

Web Builder needs to be tested in ISP-like conditions, and that means multiple applications running on a number of accounts, with as many users as possible. And these results need to be published. A single demo app, however cute, is meaningless as a deployment guideline.

TopSpeed has created a powerful web development product, but too many questions

TopSpeed
Clarion 5 by TopSpeed
FREE Microsoft Internet Explorer
etc2000 EVENT SPONSOR

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

about resource usage and scalability are going unanswered. At present, Web Builder may be a solution for complex, low usage applications, where the relatively low cost of development offsets unknown maintenance issues. High usage deployments remain an unknown risk. If TopSpeed really does want to provide mainstream web applications then published test results are in order. If not, Web Builder may be relegated to a small, niche market.

The Clarion Advisor: Browse Popup Menu Tricks

Tool Talk: Reviews News
(Mar 28,2000)

Editorial
(Mar 28,2000)

The Clarion Advisor: Browse Popup Menu Tricks
(Mar 28,2000)

Subclassing With A Twist
(Mar 28,2000)

March 2000 News
(Mar 28,2000)

by Dave Harms

Popup menus are a standard (and handy) feature of Clarion browse boxes. On any AppGen browse, you can right click to get a popup menu that corresponds to the browse's update buttons.

In ABC popup menus are managed by a PopupClass object called `*BrowseObjectName*.Popup`. This isn't immediately obvious if you look at the source for a generated browse, because you won't see any references to `Popup`. So how does the popup menu know what items it should contain?

The answer is in the browse's `SetAlerts` method - not the generated method, but the method in the parent base class. Here's an excerpt from the `SetAlerts` method in ABBROWSE.CLW:

```
IF SELF.InsertControl
  SELF.ListControl{Prop:Alrt,255} = InsertKey
  SELF.Popup.AddItemMimic(DefaultInsertName,↵
    SELF.InsertControl,'!'&DefaultInsertName)
END
```

The `AddItemMimic` method creates a popup menu item that corresponds to the insert button. There are similar calls for the delete and update buttons.

Generally speaking, if you want to modify the behavior of the popup menu you do so by changing the browse update buttons, and the menu will be created accordingly. For instance, if you disable the delete button, the delete option does not appear on the popup menu.

If you hide the one of the browse update buttons, however, the popup menu is unaffected; the user can still take action that way. You have to disable the popup item manually, and you can do this in the ThisWindow Init method, Process Field Templates embed point (among others).

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

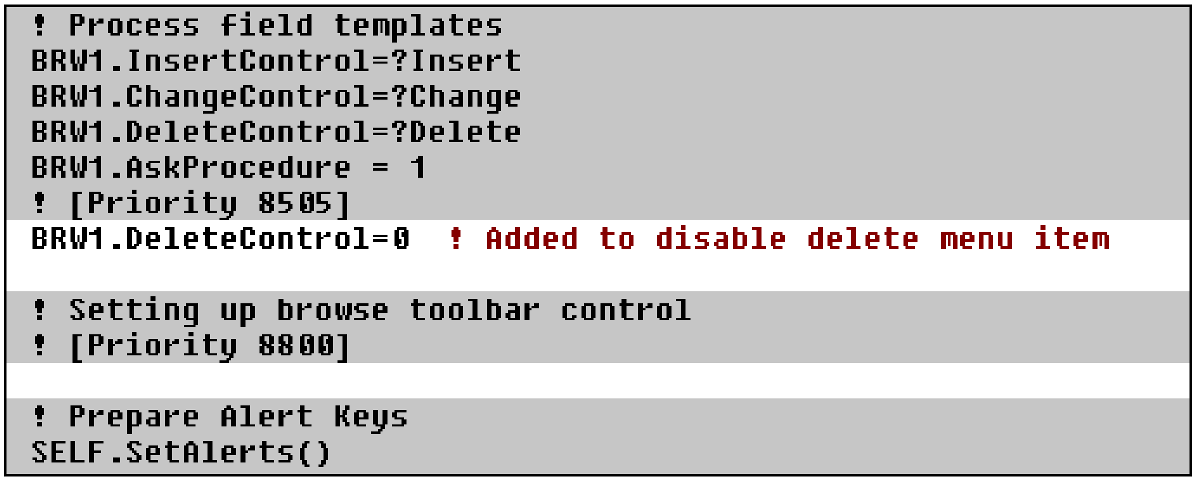
To disable the popup item for a hidden Delete button, enter the following code (assuming, of course, that the name of your browse object is `BRW1`):

```
BRW1.DeleteControl=0
```

Getting The Big Picture

I find I don't use the embed list much any more. I prefer to view the procedure source, as this gives a more complete picture of what's going on. If you're not familiar with source view, from the AppGen main window highlight the browse procedure, right click, and choose Source. Search for the text "Process field" (without the quotes). This will take you into the ThisWindow.Init method source, and you should see the generated code shown in Figure 1.

Figure 1. Generated code to initialize the browse popup menu.



```
! Process field templates
BRW1.InsertControl=?Insert
BRW1.ChangeControl=?Change
BRW1.DeleteControl=?Delete
BRW1.AskProcedure = 1
! [Priority 8505]
BRW1.DeleteControl=0  ! Added to disable delete menu item

! Setting up browse toolbar control
! [Priority 8800]

! Prepare Alert Keys
SELF.SetAlerts()
```

The generated code simply sets the browse object's public `InsertControl/DeleteControl/ChangeControl` variables to the corresponding button's field equates. I've overridden one of these, clearing the value to disable the popup menu item. Next comes a generated call to ThisWindow's `SetAlerts` method, which will in turn call the browse's `SetAlerts` method. Since there's no value for `DeleteControl` no menu item will be created.

Adding Menu Items

You can add your own items to the browse's popup menu, if you wish. The easiest way to do this to follow ABC's lead and create a button for the action to be performed, then call the popup's `AddItemMimic` method. Add this code in the browse object's `SetAlerts` method. If the code is before the parent method call, your menu items will appear at the top of the menu, and if added after the parent method call they will appear at the bottom.

For more information on popup menu methods see PopupClass in the ABC Class Library Help section.

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.*

Clarion MAGAZINE

Subclassing With A Twist

Tool Talk: Reviews News
(Mar 28,2000)

Editorial
(Mar 28,2000)

The Clarion Advisor: Browse Popup Menu Tricks
(Mar 28,2000)

Subclassing With A Twist
(Mar 28,2000)

March 2000 News
(Mar 28,2000)

by Jim Kane

If anyone out there has figured out how to leave well enough alone, please let me know. Not long ago I was asked to build an indexing system that would store a description of a scanning electron micrograph along with the actual image. The person wanting the work done had written out a reasonably complete specification. Upon accepting the job I set about building the application. About a week later I came back to set up the SQL Server and demonstrate. The customer was quite happy with it and willing to pay me on the spot to get the program. If I wasn't a fool I would have taken the money and run, and since it was a Friday I probably would have enjoyed a stress free weekend. That's if I wasn't a fool.

What actually happened is after getting the check in my hand I commented that the one thing in the specification I didn't implement, because I didn't understand it, was a requirement to show a scale above the image. I had been expecting a scale field in the database stored as a string or long that should be shown above the image, but the information was not present in the database.

The customer then explained what they wanted was basically a ruler of a sort that would scroll to show how far into the image you had scrolled. However their staff Clarion programmer had told them there was no way to do it, so they just dropped the requirement. If only I could have left well enough alone. Before I could stop my mouth it blurted out, "If you can get me the image I'd be glad to integrate it." Maybe at future meetings carrying a roll of tape to help control my mouth would be advisable.

Well, I left with the image and knew the challenge boiled down to keeping the two image controls – one with the image and one with the ruler - in sync. Putting that in Clarion terms, what I needed was:

```
?HRulerFEQ{Prop:Xorigin} = ?ImageFEQ{prop:Xorigin}
```

If I could maintain that relationship then the two image controls would always be in sync. In order to make it look better, and to avoid users scrolling the ruler image independently of the main image, I also needed to hide the scroll bar on the ruler but still have a scrollable image. Lastly, there was often more than one main image control on the screen at once so I needed to be able to have multiple instances of the code running. I suggest you download the code that accompanies this article and run the sample program. I think that will make what I'm trying to explain much easier to understand.

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

The one thing left out of the simple line of code above was when should the code be executed? The obvious answer is whenever `prop:hscrollpos` of `?ImageFEQ` changed. Unfortunately there is no Clarion event called `Event:HeyKaneHscrollPosJustChanged`. What I needed to detect is when the Windows messages `WM_HSCROLL` and `WM_VSCROLL` happened. And that is what subclassing is all about!

Subclassing

Subclassing is the ability to intercept Windows messages before they are processed. (It also has a meaning for object-oriented programming but that's not the subject of this article.) This gives the programmer the power to decide whether or not to pass on the Windows message to the Windows function that would normally act on the message. For example, if Windows detects a mouse movement over a control it sends that control a `WM_MouseMove` message. Through subclassing it is possible to detect the mouse movement and optionally prevent the control from getting, and therefore acting on, the information. Virtually everything Windows does is through messages. To say the ability to tap into and potentially alter the message stream is a powerful tool is a bit of an understatement. Hopefully you will only use it for good!

Just about anything can be subclassed as long as the operating system (OS) thinks it is a window and assigns it a window handle, or `hwnd`. `Hwnds` are easy to obtain in Clarion:

Property	Description
`?yourcontrol{prop:handle}`	`hwnd` of your control
`0{prop:handle}`	`hwnd` of the current window
`0{prop:clientHandle}`	`hwnd` of the client portion of the current window

Combined with `SetTarget()` the above make it very easy to get the `hwnd` for virtually any window in an application. In fact, it is possible to subclass the application frame, any window, just the client portion of the window (the area of the window inside the frame, border, menus and toolbars), or an individual control. Where to subclass depends on the message to be monitored or intercepted.

Unfortunately, sometimes the only way to determine what to subclass is to try a few different things until you find the message of interest to you. For example, for mouse tracking over a control subclass the control, for mouse tracking over the client area of the window subclass the client area of the window, for mouse tracking over the window frame subclass the entire window.

Sometimes this division can be a real problem. If a control is subclassed and mouse movement over the control is being detected, when the mouse leaves the control and goes out over the client portion of the window or leaves the window, no "hey I'm leaving" message is received by the control. As a result, it is often desirable to subclass both an individual control and the client area of the window so mouse movement out of the control can be easily detected. There is another option called mouse capture, but it has some potentially nasty side effects so I'm going to just mention it but not explore it further.

Subclassing only requires three steps.

Subclassing Step One

Obtain the address of the OS's message handling procedure for the subclassing target and substitute the address of your custom message handler. The OS's message handling procedure is typically called the `WndProc`. Clarion makes getting this address easy. Here is the code for three different subclassing targets: you just pick the one you want to use.

```
!address of a Window's WndProc
AddressOfWndProc = 0{prop:wndProc}

!address of a Window's client area WndProc
AddressOfWndProc = 0{prop:ClientWndProc}

!address of a control's WndProc
AddressOfWndproc = ?yourcontrol{prop:wndProc}
```

Then substitute the address of your custom procedure (remember, you only need one of these three):

```
0{prop:wndProc}=address(YourCustomSubclassProcedure)

0{prop:clientProc}=address(YourCustomeSubclassProcedure)

?YourControl{prop:wndProc}= |
  address(YourCustomSubclassProcedure)
```

Alternatively, and with exactly the same functionality, the API can be used to do the two steps above in one step:

```
AddressOfWndProc =  SetWindowLong(0{prop:handle} |
   ,-4,address(YourCustomSubclassProcedrue))

AddressOfWndProc = SetWindowLong(0{prop:clienthandle} |
   ,-4,address(YourCustomSubclassProcedrue))

AddressOfWndProc = SetWindowLong(?YourControl{prop:handle} |
   ,-4,address(YourCustomSubclassProcedrue))
```

The –4 is just one of those Windows constants (`GWL_WndProc`) taken from a C header file.

> **Editor's Note:** Since this article was first published TopSpeed's Alexey Solovjev has indicated that `SetWindowLong` can cause problems with the Clarion runtime library, and is therefore not recommended. An official statement to this effect is expected at some future time.

Subclassing Step Two

Next you need to write a subclass procedure to do the custom message handling. This is the fun part, where you get to take revenge on Windows. Windows may be expecting to receive certain messages but if the programmer suppresses them, Windows does not get them – the programmer is in control!

The subclass procedure must have the following prototype:

```
SubClassProc procedure(unsigned hwnd, unsigned msg, ↵
   Unsigned wparam, long lparam),long,pascal
```

Please note the prototypes in the language reference manual under `prop:WndProc` are only correct for 16 bit whereas the prototype above will work for both 16 bit and 32 bit. Also note the required `pascal` calling convention.

Because this specific prototype is a Windows requirement, a class method may *not* be used. All Clarion classes have a first parameter of `SELF` and that violates the required prototype above. In addition, because all the custom event handling code is off in a separate procedure, it can be awkward to integrate with the rest of the program since local variables in the mainstream of the program are not visible (out of scope) in the subclass procedure.

At first, for my project, I thought that to subclass several image controls I'd have to write an equal number of subclass procedures. That would also mean if I changed the code in one I'd have to make like changes in the others. In addition, there were multiple different windows, each with several image controls. This was beginning to sound like work! But fortunately my hatred for repetition and my fear of not keeping multiple copies of code in sync stepped in and forced me to invent. More on that later. I know, you can hardly contain your excitement, but be patient. It's coming.

Subclassing Step Three

When the custom message handler finishes execution, there are two options. One is to simply return a 1 to suppress further processing of the message. In most cases this tells Windows that the message was processed, and nothing else be done. More commonly you will pass the message on to the normal window message handler or `WndProc`. Since you saved the address of the `WndProc` in step one, just call that address, passing on the parameters the custom message handler received. This is done with the Windows API function `CallWindowProc`():

```
Result = CallWindowProc(AddressOfWndProc, ↵
   hwnd, msg, wparam,lparam)
```

Please note the last four parameters are the exact parameters passed to the subclass procedure. The first parameter is the address saved in step one.

A Quick Example

Add the following code to your application to get subclassing working. In the Global Map embed:

```
  SubClassProc(unsigned, unsigned, unsigned,long),long,pascal
  Module('api')
    CallWindowProc(long,unsigned, unsigned, unsigned,long),↵
      long,pascal,name('callwindowproc')
  end
```

In the Global Data embed:

```
! store addressOf WndProc
Savedproc1 long

! field equate label or FEQ of image control
ImageFEQ long

! FEQ of the horizontal ruler image to be
!  scrolled in sync with ImageFEQ
ImageRulerH long

! FEQ of the vertical ruler image to be scrolled
!  in sync with ImageFEQ
ImageRulerV long
```

In the After Opening Window embed:

```
Open(window)
Savedproc1 = 0{prop:wndProc}
0{prop:WndProc}=address(subclassproc)
!save feq of the image control
ImageFEQ = ?image1
!save feq of the ruler image control we want to scroll by magic.
HRulerFEQ = ?HRulerFEQ
!save feq of the ruler image control we want to scroll by magic.
VRulerFEQ = ?VRulerFEQ
```

In the Program Procedures embed:

```
SubClassProc procedure(unsigned hwnd,unsigned msg,↵
    unsigned wparam,long lparam)

  Code
  !custom message handling
  if msg=114h !wm_hscroll
     HRulerFEQ{Prop:Xorigin} = ImageFEQ{prop:Xorigin}
  End
  If msg=115H !wm_vscroll
      VRulerFEQ{prop:Yorigin} = ImageFEQ{prop:Yorigin}
  end
  return callwindowproc(savedproc, hwnd, msg, wparam, lparam)
```

This works, but imagine the case I described earlier where there may be many subclassings going on at one time. Tracking all those procedures and all those global variables does not sound like a fun way to spend a weekend. Additionally, for each different subclass, I would need a different `ImageFEQ`, `HRulerFEQ`, `VRulerFEQ`, and `Savedproc`.

When I saw that I started to think (dangerous, I know, but sometimes it helps) that maybe I could put the four `FEQs` in a queue along with the Windows `hwnd` that identifies the window. I cleverly named the queue `HwndQ`. When the subclass procedure is called it receives hwnd, so I could do a `Get()` on the queue by that value.

It sounded like a plan. It had a fatal flaw, but then again most of my plans do. Fortunately I didn’t know that so I went on. I moved the "After opening the window" code to a class init method; it looked like this:

```
Rulercl.Init   Procedure(window pWindow, long pImageFEQ)
  Code
   Clear(hwndQ)
   HwndQ.hwnd =pWindow{prop:handle}
   HwndQ.ImageFEQ = pImageFEQ
   HwndQ.SavedProc1 = pWindow{prop:wndProc}
   !code to create ruler image controls on the
   ! fly and provide FEQs
   HwndQ.HRulerFEQ =HrulerFEQ
   HwndQ.VrulerFEQ =VrulerFEQ
   Add(hwndQ, hwndQ.hwnd)
   PWindow{prop:WndProc}=address(subclassproc)
   Return
```

Note that the `HwndQ` is not a class member (no `SELF`) but rather declared globally in the class's source (CLW) module. This is because all instances of the class will store their data in the one queue.

Likewise I moved the subclass procedure out of the application map and put it in the class's CLW file:

```
SubClassProc Procedure(unsigned hwnd, unsigned msg, ↵
   unsigned wparam, long lparam)

  Code
  !hey Windows provides this - use it!
  HwndQ.hwnd = hwnd
  Get(hwndq, hwndQ.hwnd)
  !bail out if you can't find hwndQ.savedproc1
  If errorcode() then return 1.
   if msg=114h !wm_hscroll
     HwndQ.HRulerFEQ{Prop:Xorigin} |
       = HwndQ.ImageFEQ{prop:Xorigin}
   End
   If msg=115H !wm_vscroll
     HwndQ.VRulerFEQ{prop:Yorigin} |
       = HwndQ.ImageFEQ{prop:Yorigin}
   end
   return callwindowproc(hwndq.savedproc1, hwnd |
    ,msg, wparam, lparam)
```

The one thing worth noting in the revised code is if an entry is not found in the `hwndQ`, then hwndQ.savedproc1 isn't valid. If you ignore that and execute `CallWindowProc` with a bad first parameter the chances of a GPF approach 100%. So in that circumstance simply return a 1 which is always safe.

Well at this point, I was darn proud of myself. By adding my class to every procedure that needed ruler support, and adding an `Init` and `Kill` method call (see the code in the zip file for the boring `Kill` code), I could get a ruler to appear, and it scrolled with the main image. And it only took two lines of code.

At that point while I was feeling good my wife walked in and wanted to know if I could pry myself away from that (bad word deleted) computer long enough for a family outing tomorrow. Since things to that point were going well, I mustered my cockiest voice and said, "No Problem!"

Unfortunately no sooner did she leave the room than my least favorite physician, Dr. Watson, magically appeared. In my original test case, I only had one image control on the window. When the `Init` code for the second control executed, the good doctor popped up.

After a little thought, mostly about the pain that would have been inflicted if I had to go back on my word and didn't attend tomorrow's family outing, I realized one of the dangers of using a global `HwndQ` was that, while one image control is initializing, any previously running image controls may be processing scroll messages and doing a `Get()` on the `HwndQ`. This is a typical side effect of global variables and why I avoid them, but in this case I had to use a queue that spanned all the instances of the class, and that meant global.

What I needed was some kind of traffic cop so that while the `Init` code was running and trying to add a queue record, the queue was not also being accessed in the subclass procedure. Those of you familiar with multi-threaded programming may be thinking about wrapping `HwndQ` in something like a critical section. While I wouldn't argue with that, I had an important deadline approaching so I opted for a simple solution. In the subclass CLW file, I added yet another global variable, `RulerInitializing`. At the start of the Init method code I added this code:

```
RulerInitializing = true
```

And just before the return at the end of the `Init` code I added:

```
RulerIntializing = false
```

The net result is while the queue is being set up, `RulerInitializing` is set to `True`. That is the signal that the subclass procedure, which may be called from another procedure or control, is not allowed to mess with the `HwndQ`. In the subclass procedure I added a bit of code to test for the condition:

```
SubClassProc Procedure(unsigned hwnd, unsigned msg ↵
   , unsigned wparam, long lparam)

  Code
  !bail out if hwndq is in use.
  If rulerintializing then return 1.
  !hey Windows provides this - use it!
  HwndQ.hwnd = hwnd
  Get(hwndq, hwndQ.hwnd)
  !bail out if you can't find hwndQ.savedproc1
  If errorcode() then return 1.
```

That one line protected me from using `HwndQ` while initializing, and cured the GPF.

There you have it. By putting subclassing into a class you can run as many instances of the subclassing code in your application as you wish without multiple copies of the subclass procedure code or global variables to manage. Also all the prototypes are nicely tucked into the class, and don't have to be retyped. And perhaps most importantly, I got to go on the family outing. Oh, and I also made my customer so happy he has buried me in additional work.

Epilogue: Life In The Real World

The ruler bitmaps in the sample application may not display correctly on your screen. They were made on my laptop with a display at 120 dots per inch. As a result, one inch on the ruler will only actually measure one inch if your screen happens to be 120 dpi. To avoid that problem, the best thing to do is to generate the images on the fly at runtime for the video resolution detected. This way you can also customize dimensions for any nationality or scale needed. For example, for the project I was working on, various scales of microns were needed.

Download the source

Jim Kane was not born any where near a log cabin. In fact he was born in New York City. After attending college at New York University, he went on to dental school at Harvard University. Troubled by vast numbers of unpaid bills, he accepted a U.S. Air Force Scholarship for dental school, and since graduating has served in the US Air Force. He is currently the Officer in Charge of Dental Facility Design at USAF Dental Investigation Service in San Antonio, Texas. In his spare time, he runs a computer consulting service, Productive Software Solutions, which he hopes to run full time after retiring from the US Air Force Dental Corps in June 2000. He is married to the former Jane Callahan of Cando, North Dakota. Jim and Jane have two children, Thomas and Amy.

Clarion MAGAZINE

Clarion News

Tool Talk: Reviews News
(Mar 28,2000)

Editorial
(Mar 28,2000)

The Clarion Advisor: Browse Popup Menu Tricks
(Mar 28,2000)

Subclassing With A Twist
(Mar 28,2000)

March 2000 News
(Mar 28,2000)

Read The February 2000 News

March 28, 2000

Clarion 5.5 B2 At Duplicators

C5.5 Beta 2 is now at the duplicators, and should be shipping in a couple of days. TopSpeed Sales: 800-354-5444 or 954-785-4555 or contact them via e-mail.

Ragazzi Leaves Third Party Market

As of April 1st, 2000 Software By Ragazzi is leaving the third Party Clarion For Windows arena. The complete source of the Templates, Utility Library and Developer's toolkit will be available for download to all registered users. The source is being provided as is. Email FullSource@Software-By-Ragazzi.com to receive the password and location of the download file.

Clarion MS SQL Discussion List

JVZ Systems CC has set up a discussion mailing list for Clarion develoeprs who use the MS SQL file driver.

BackFlash 4.1 Released

Sterling Data has released BackFlash version 4.1. New in this release: customizable backup completion message and WAV, and option to disable disk space test. Demo available.

March 21, 2000

Clarion 5.5 Beta 2 Imminent

Rumor has it that B2 has been mastered and will ship shortly.

Whitemarsh Website Update

The newest metabase version (beta 3.03) has finally been uploaded to the website. During March, Whitemarsh will be working at completing the gold version of the metabase that will include significant additions for Schema DDL export and import, and other programs for data model reverse engineering.

LSZip for Clarion 5.5 Beta 2

LSZip 2.5 Data Compression Library (release date: 07-Mar-2000) is fully compatible

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

with the Clarion 5.5 Beta 2 final install and available for download from Linder Software. This update is free to registered users. Because some serial numbers and installation keys are available on various "warez" sites, the old installation key is no longer valid. Current LSZip users should contact Linder Software at sales@lindersoftware.com us for an updated password.

NetTalk 1.0 Beta 5 Released

NetTalk 1.0 is a toolset that allows your programs to communicate with each other over TCP/IP networks. The focus has been on ease of use for the developer, and simplicity of deployment. New in this release are increases in speed and robustness. NetTalk will usually cost $299, but is $199 for the duration of the beta program. For more information see www.capesoft.com.

Secwin 3.0 Beta 5 Released

This should be the final beta before Secwin 3 goes gold. New features include SQL and product registration enhancements, expiry date warnings, and control restrictions.

TearOff 1.5 Released

TearOff allows your user to create a dockable toolbar by Ctrl-Clicking on existing menu items. Version 1.5 fixes known bugs and adds some new features.

Special Agent 1.2 Released

Special Agent allows you to build the Microsoft Agent OCX straight into your Clarion program. Version 1.2 has an improved kernel, which results in more consistent behaviour, and no more 'real-funnies' that sometimes appeared before. Entry Control support has also been improved, and a version for Clarion 5.5 is also now available. The normal price for Special Agent is $199, but is on sale for $179 until April 15th.

Status Bar Control

Leonid Chudakov has uploaded a Status Bar Control to his web site. This control lets you put icons and/or a progress bar inside the status bar. Buy the TX wrapper and get the source code of the all common controls for free.

Clarion Handy Tools N-8 Build For C5.5 Beta 2

Clarion Handy Tools for C55 Beta2 are now available for download to all registered users. Use your FTP download wizard as usual. The DLL version install is called hndtlsndll_c55Beta2.exe, and the source code version install is called hndtlsnsrc_c55Beta2.exe.

Clarion 5.5 Professional Edition Pricing Change

The upgrade price for C5.5 Professional Edition has been reduced to $350, and will no longer contain the Internet functionality available in Enterprise edition. This change is intended to benefit developers who have no need for ‘net development. You can reach TopSpeed at 800-354-5444 or 954-785-4555.

DBWired Partners With TopSpeed

DBWired, an ASP/ISP, has partnered with TS to deliver Web Edition ISP services. DBWired has become a new ISP partner. As an introductory special, DBWired is offering 30 megs of Web space for $24.95, which includes start-up fees. E-mail eschuler@dbwired.com for more details.

Evaluation CDs Discontinued

TopSpeed has discontinued the Evaluation Edition of Clarion in favor of a 30 day trial purchase. The evaluator's credit card will be charged only for shipping if the product is returned within 30 days. Call TopSpeed Sales at 1-800-354-5444 or 954-785-4555. You can also reach them via e-mail.

March 14, 2000

UltraTree Platinum New Release

Paragon has released the Tagging and Reporting feature of UltraTree Platinum. Features include row, branch, and tree tagging/untagging, auto-advance, and tagged reporting. The Tagging and Reporting feature is a premium feature, available only to Platinum premium support plan subscribers, or by separate purchase to Platinum standard support users. Standard features in Platinum include page-ranging, multicolumn trees, recursive trees, and more.

Create COM Components In Clarion

Jazz Age's JA Objects Edition 1.7.103 for Clarion is freely available for download. This product makes it possible to create COM components with full ASP and MTS compliance, and dual interfaces. Advanced COM threading models are supported.

PowerRUN Templates

Greg Berthume's PowerRUN templates pick up where RUN() left off. Use them to: launch programs hidden, minimized, in a window or maximized, with or without focus; run DOS/Window processes in the background, waiting (or not) until termination; set the priority in 32bit; load websites, documents and files via user's default web browser or application via ShellExecute; play WAV sound files. PowerRUN works in 16 and 32bit, Clarion 2003 and earlier, C4,C5,C5.5 ABC/Legacy, and is multi-DLL compliant and DET compatible. Price: $13.

SETI@Home
Clarion Developers For SETI

Convinced there's no intelligent life on this planet? Start looking elsewhere. Put your computer to work in its idle moments, analyzing data from SETI, the Search For Extraterrestrial Intelligence. SETI@home provides a free screensaver utility that crunches radio telescope data while your system is idle. The program has options for those who have dial-up accounts to either request a connection when needed or to automatically connect when needed. A constant connection is not required.

etc2000: Ten Weeks And Counting

The East Tennessee Clarion Conference & Gathering, take two, is just 10 weeks (70 days) away, and counting. Three and a half days of in-depth sessions, reception dinner and hors d'oeuvres, three full breakfasts, lunches, and dinners, including the second Cajun Cookout. Presenters include James Fortune, Dave Harms, Nik Johnson, Steve Parker, Skip Williams, and Andy "Cowboy" Stapleton. All this for $369.00. For the small additional charge of $50.00 you get to heckle Mr. Bruce Johnson in his full day class Saturday, May 27. So far attendees are coming from all around the US, Canada, United Kingdom, South Africa, Australia, Norway, The Netherlands and Denmark. We hope to see you in East Tennessee in May.

Perl/Html Templates

The Open Linux Project for TopSpeed is looking for template programmers who want to expand their knowledge of generating code for other languages. This project involves creating CGI templates which use Perl to generate HTML.

March 7, 2000

C4 Medical Billing Source Code Available

For Sale: MedProWare Medical Billing Software (C4 Source Code) with online help. Program in use for 2 years with no reported bugs in last six months. Price negotiable.

Simsoft Templates Include Calculator

The Simsoft Templates now have a calculator that attaches itself to an entry field either when the field is selected or optionally when a button is pressed. Demo is available at www.clarionshop.com.

G-Buddy Released

Gitano Software's G-Buddy, "the best Clarion friend you'll ever have" includes five applets designed to make the developer's life a little easier. Keep track of and edit images, templates, text files, color changes, message boxes, and much more, all easily viewed and edited. More info and download at

G-Reg 4.01 Released

This new release of Gitano's G-Reg has added features such as limited number of users on a network, registration keys, hardware ID, an extensive help file, and more.

VooDoo For Clarion Demo

A VooDoo Clarion demo is now available for download. VooDoo is a graphics library that plays multimedia files, blits, pans, scrolls, scales, and rotates images, handles hidden surfaces and sprites, and performs a variety of other graphics functions. Joysticks, text-to-speech, TWAIN devices and more are also supported.

Handy Tools Update

The Handy Tools downloads page has been updated. All demo applications and free software are now available individually from the downloads page using HTTP. The FTP wizard is still available. New features in Handy Tools include registry functions, FTP class changes, free compile manager, free image manager, and an unlimited FTP demo application,

LSZip 2.50 Free Update

The brand new LSZip 2.50 Compression Library will be available March 7. The new LSZip 2.50 now supports high-performance in-memory compression, callbacks, uuencode and uudecode, a new LSZip ABC Class, new source code examples, a new Programmer's Reference Manual in PDF format, and more. This update is free to registered users.