

Clarion MAGAZINE

[Search](#)

[Home](#)

[COL Archives](#)

[Subscribe](#)

[New Subs](#)

[Renewals](#)

[Info](#)

[Log In](#)

[FAQ](#)

[Privacy Policy](#)

[Contact Us](#)

[Downloads](#)

[PDFs](#)

[Freebies](#)

[Open Source](#)

[Site Index](#)

[Call for](#)

[Articles](#)

[Write For Clarion Magazine](#)

Clarion Magazine is looking for writers! I want to expand the magazine's coverage of Clarion programming issues, and I know there's a lot of talent out there just waiting for a chance to shine.

Posted Thursday, April 26, 2001

[The Weekly PDFs Are Back!](#)

For those of you who like to download your issue for offline reading, the weekly PDFs are back! Also available: the April 2001 PDF.

Posted Thursday, April 26, 2001

[The Clarion Advisor: Just In CASE](#)

This week's programming tip comes from Bruce Johnson at CapeSoft, who points out that the CASE statement is a lot more flexible than many developers realize.

Posted Tuesday, April 24, 2001

[Product Review: VariView 2.0e From Solace Software](#)

Wouldn't it be nicer if there were an easy way to build a window into your program that simply let you view the contents of any variable or file buffer? On demand? Without impacting your program flow or requiring adding code, removing code and constantly recompiling? Is this a series of leading questions, or what?

Posted Tuesday, April 24, 2001

[Clarion and Multi-Edit: Together At Last](#)

Clarion's own source code editor, although functional, isn't as feature-rich as some other programmers' editors, like American Cybernetics' Multi-Edit. Multi-Edit integrates with any popular programming tools, and as Vince Du Beau explains, you can use Multi-Edit

[New Clarion Web Server In Development](#)

[PD Lookup Update](#)

[OpenClarion Site Rebuilt](#)

[CPCS PDF/Emailer Addon enhancements](#)

[PD Translator Plus Supports Ideographic Characters](#)

[Linder Software's WebUpdate\WebSetup Beta Coming Soon](#)

[New ExceleTel TeleTools Samples Available](#)

[DC Templates v1.7](#)

[ARCO Word Reporter XP Compatible](#)

[Greenbar Template Gets New Features](#)

[Free Greenbar Template](#)

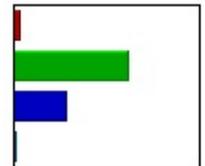
[Solace ColourAll Template Now Free](#)

[Free Editor Key Configuration Tool](#)

[Handy Tools Build O-](#)

SURVEY

Which browse & form control method do you use?



Toolbar
 Button
 Both
 Neither

Vote s: 71

with Clarion too.

Posted Tuesday, April 24, 2001

No Mag This Week; Instead, A Book

I had originally planned to publish several articles this week, as usual, but instead I've been busy preparing a support web site for my new book, titled [JSP, Servlets and MySQL](#).

Posted Tuesday, April 17, 2001

Is SQL A Real Standard?

In this Whitemarsh paper, Mike Gorman takes a look at the SQL "standard." If a standard is something you can count on, for what it means, says and does, then perhaps SQL really doesn't qualify anymore.

Posted Sunday, April 15, 2001

Writing Classes That Create Word Documents

It's easy to create Word documents from a Clarion application, using Jim Kane's OLE classes. Dave Harms looks at the OLE code, and ponders a set of Word-specific classes.

Posted Tuesday, April 10, 2001

SetFilter to the Max

In the last action packed episode Steve Parker examined the use of SetFilter in some depth. Now it's time to apply that knowledge and use SetFilter directly.

Posted Tuesday, April 10, 2001

Clarion and the Internet:

Publishing Static Data

Publishing data on the Internet is easy! Just ask Tom Ruby - he's been doing it for years. As Tom explains in this article, the trick is to use the Clarion HTML driver...

Posted Tuesday, April 03, 2001

Dynamic Filters: Applying The Theory

Dr. Parker, having dealt with the theory of dynamic filters, goes on to some sample code, which ranges from the sublime to the ridiculous.

Posted Tuesday, April 03, 2001

April 2001 News

Clarion news for April, 2001.

Posted Sunday, April 01, 2001

[6A Now Available](#)

[Linder SetupBuilder](#)

[3.51 \(Build 6084\)](#)

[Available](#)

Copyright © 1999-2001 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca

Clarion MAGAZINE

The Clarion Advisor: Just In CASE

Published 2001-04-24

This week's programming tip comes from Bruce Johnson at CapeSoft, who points out that the CASE statement is a lot more flexible than many developers realize. For instance, if you're evaluating a variable against a number of different conditions, you can end up with code that looks something like this:

```
IF cus:type = 1 OR cus:type = 9 OR |
    cus:type = 13 OR cus:type = 22
    ! do something
END
```

That kind of code gets messy, which makes it difficult to read and maintain. Ideally, Clarion would let you do something like this:

```
if cus:type = 1 or 9 or 13 or 22
    ! do something
end
```

That code won't compile, but you can do something very similar with CASE and the OROF operator:

```
CASE cus:type
OF 1 OROF 9 OROF 13 OROF 2
    ! do something
END
```

That's not all – you can also check ranges with the TO operator:

```
CASE cus:type
OF 1 OROF 9 OROF 13 OROF 2 OROF 100 TO 200
    ! do something
END
```

Thanks to Bruce Johnson for this week's tip. If you have a programming trick or technique you'd like to share, send it to editor@clarionmag.com.

[Search](#)

[Home](#)

[COL Archives](#)

[Subscribe](#)

[New Subs](#)

[Renewals](#)

[Info](#)

[Log In](#)

[FAQ](#)

[Privacy Policy](#)

[Contact Us](#)

[Downloads](#)

[PDFs](#)

[Freebies](#)

[Open Source](#)

[Site Index](#)

[Call for](#)

[Articles](#)

Reader Comments

[Add a comment](#)

It is probably worth adding that ELSE was added to the CASE...

Copyright © 1999-2001 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca

Reborn Free**CLARION**
*online*published by
CoveComm Inc.

Clarion MAGAZINE

Product Review: VariView 2.0e From Solace Software

by Tom Hebenstreit

Published 2001-04-24

There are basically two kinds of programming errors: Those that crash the application, and those that screw up your data without crashing the program.

Of the two, the second can be much more insidious in that you *think* your program is working fine when it really isn't doing what you want it to. And what are your options in that case? Well, if you are *really* bold, you use the built-in Clarion debugger to track the contents of the variables or file buffers in question.

Or, if you are like the rest of us, you start scattering `message()` statements throughout your code to try and determine where things are going wrong. The problem with using `message()`, though (apart from the fact that it is tedious, time-consuming and brings your program to a screeching halt every time one pops up), is that you also have to correctly guess when, where and what to display.

Wouldn't it be nicer if there were an easy way to build a window into your program that simply let you view the contents of *any* variable or file buffer? On demand? Without impacting your program flow or requiring adding code, removing code and constantly recompiling? Is this a series of leading questions, or what?

Well, the answer to that last question is a definite yes. And the folks at Solace Software think they have come up with a product that also answers the rest of those questions with a resounding yes. Named VariView, it is a template set that allows you to add the following capabilities to your programs:

- View the contents of any variable or file buffer, including those declared by hand in embed code.
- Log events as they happen along with the control that generated the event.
- View details of the operating environment (e.g., amount of memory, available drives, the operating system version,

[Search](#)[Home](#)[COL Archives](#)[Subscribe](#)[New Subs](#)[Renewals](#)[Info](#)[Log In](#)[FAQ](#)[Privacy Policy](#)[Contact Us](#)[Downloads](#)[PDFs](#)[Freebies](#)[Open Source](#)[Site Index](#)[Call for](#)[Articles](#)

- etc.).
- View the status of all files (open, closed, the number of records).
- Log the procedure call chain so that you can see who is calling who.
- Watch specified variables for changes. The templates record which event and control caused them to change.
- Capture and store messages in a queue for later viewing (avoids stopping your program with standard `message()` statements).
- Save all of the captured information to a TPS superfile. This can be very useful for remote debugging, as the user can just send the file back to you (the developer) and you can see exactly what was going on.

Now at version 2.0e, VariView has gone through quite a few rapid updates as Solace has added new features and fixed the occasional nit. For example, during the course of writing this review I went through versions 'c', 'd' and now 'e'. Be sure to check their web site (listed at the end of this review) to see the most current feature list.

Installation

The VariView installer is rather simplistic. It doesn't default to the actual location of your Clarion installation, so if you aren't installed in a C:\C55 folder, you need to browse for the proper folder. When browsing, it uses the standard (and confusing) dialog saying that the templates are going to be installed in the folder you select (e.g., E:\C55), when it really means that it will install the files in the proper folders *underneath* the folder you chose.

Earlier versions of the install also neglected to mention where the VariView documentation was being installed, necessitating a folder search before you could even get started. In response to a suggestion of mine on that front, Solace now displays a message at the end of the install informing the user that the help file is in the C55\bin folder. I'd like to see that taken one step further, and have an option to view the help file at the end of the install. I've always found that to be a great way to get oriented with a product right out of the box (so to speak).

After the install has completed, you have to manually register the templates.

Implementation

Adding VariView to an application is pretty easy - assuming you follow the step-by-step instructions carefully. Initially, I blundered on through and had some strange results before realizing that I had accidentally missed a step (doh!). Once I straightened that out, I found that I had no problems setting up VariView in either a simple EXE or a large multi-DLL application.

Basically, using VariView requires that you add its global extension template to the application, and then you add the two core VariView procedures (the Viewer popup window and one that handles the VariView queues). Adding the procedures is easily accomplished simply by importing a TXA file provided by Solace. This is a nice touch, as using a TXA instead of a procedure template means that you can modify the look and feel of the VariView viewer window to match your own program's look and feel.

If you are creating a multi-DLL application, the VariView procedures should be imported into your data (dictionary) DLL. You must also add the global extension to every other DLL and the EXE, setting a few template prompts as needed. Once you have added the global extension, a VariView procedure extension is automatically added to every other procedure in the application. It is those procedure extensions that allow you to tailor VariView for each procedure's individual needs (or just disable it for a given procedure).

Earlier versions of the templates required that you also add the two VariView procedures to each DLL and the EXE as external procedures. This was tedious in my eleven-DLL test application, but not a killer. The latest release removed that requirement; the templates add the VariView procedures transparently. Once I went back and removed all of the external references that I had added earlier (sigh), everything worked as expected with the new version.

VariView allows a fairly wide range of options for specifying just exactly what it should display or ignore. You automatically get all file buffers and variables that are declared in 'Data' buttons (both global and local). VariView also offers a number of optimization options, as shown below in Figure 1.

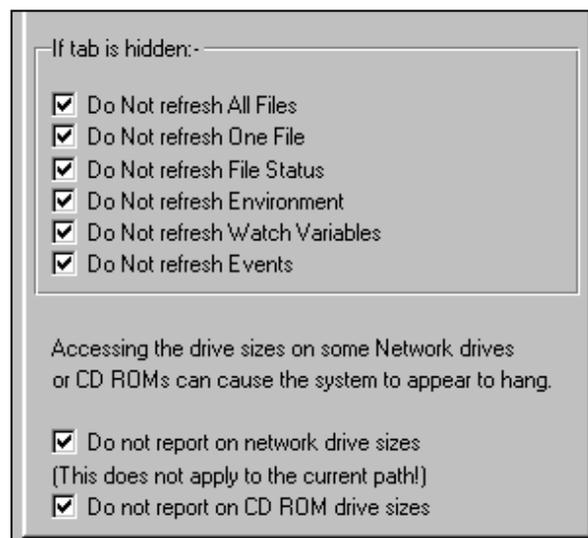


Figure 1. Some of the options for optimizing VariView

As you can see, these options let you suppress actions for tabs in the VariView viewer window that are not currently visible.

You can manually add variables or object properties that are declared outside the view of the template generation system, such as variables declared in embed code. You can also globally specify that VariView ignore certain files and variables. I found this feature particularly useful where I had declared an object as a TYPE in a data button. VariView was trying to treat the TYPE as a variable, and the compiler didn't like that one bit. By adding the TYPE to the procedure's VariView ignore list, I easily solved the problem. You can also ignore specific events, such as a timer that would otherwise flood the VariView events queue.

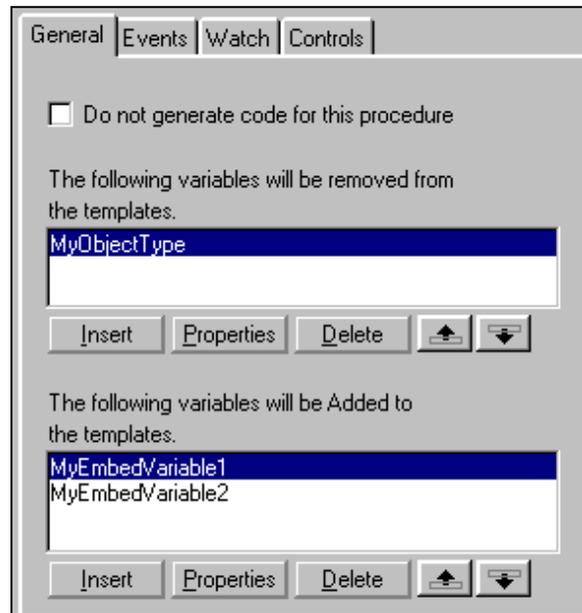


Figure 2. A condensed view of the VariView Procedure extension General tab

You can also see that it only takes one checkbox to stop VariView from generating any code for the procedure.

I did run across one other problem that I don't believe to be Solace Software's fault. On some of my data entry forms, I display a few object properties as string values. I found that the compiler choked on the Solace-generated code if the property labels were entered using dot syntax, e.g., `MyObject.MyProperty`. The reason was that Solace would create an equate label to pass to the viewer like this: `?MyObject.MyProperty`. Changing the separator from a dot to a colon made the compiler happy again, e.g., `?MyObject:MyProperty` worked just fine.

All in all, setting up VariView was quite painless – especially the newer 'e' release.

Performance

When I first tried to use VariView, I experienced some strange behavior where I couldn't view the contents of my file buffers. A quick trip to the help file, however, revealed the source of the

problem: my application frame had a timer that was displaying the system clock in the message bar. As a result, my application was constantly swapping from the thread where my browses and forms were back to the application frame thread. Since my files buffers were threaded, this had the effect of blanking out the buffer contents in VariView after a half second or so.

Solace provided two workarounds – I chose to just disable the VariView code for the frame (a simple checkbox) and then everything worked as expected.

It was really neat to be able to inspect the contents of my buffers and variables as the program ran. For example, VariView immediately flushed out a problem where a variable file name was not being initialized correctly. It was also trivial to track down a case where the contents of a variable were being truncated – just by watching where and when the variable changed, I found that an intermediate location where it was being stored was too small. I also found it easy to checking that file buffers were initialized properly on adds, and to verify that that parameters were being set up correctly in wizards.

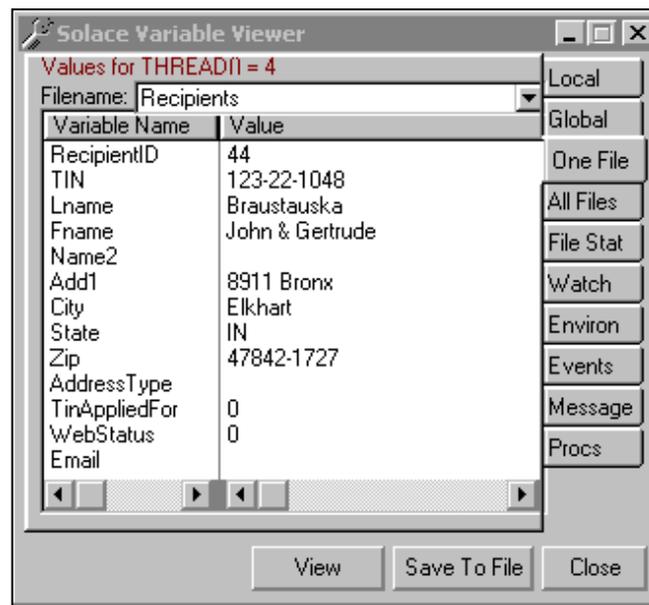


Figure 3. The VariView viewer window, showing the contents of a file buffer

Note all of the tabs to the right of the browse in Figure 3. These should give you some idea of the number of options that VariView offers for your viewing and debugging pleasure.

What about cases where buffer contents are being rapidly changed, e.g., during a process or report? VariView has that covered as well. You can use a code template to send a stream of messages to VariView, which will add the messages to one of its queues so that you can review them later (the 'Message' tab shown in Figure 3). And that is *far* better than using messages and having to click OK for each report record, etc. You can also incorporate the message logging calls in your own embed code, as

shown by a simple example in the help file.

Running on a 500Mhz Pentium III, I didn't find that having the VariView window active caused any noticeable slowdown in the execution speed of my programs. If the VariView window is not open, the VariView code is not executed at all (it is all wrapped in a big IF test.)

Size-wise, VariView added approximately 210k to a 1,261k application, which I don't consider unreasonable given the functionality VariView adds.

Documentation

Documentation for VariView is provided in the form of a standard Windows help file. The help is reasonably thorough, and does give step-by-step instructions for adding the VariView templates to your applications.

One glaring omission is that I could find no mention of how to contact Solace Software in case you need technical support. Web and email contact information should always be featured prominently in any help file, in my opinion.

Technical support

Support is provided via email. Solace Software is also active in the SoftVelocity third-party newsgroup. The messages I sent to Solace were answered promptly and they were very agreeable to suggestions for improving the product.

Room for improvement

As mentioned above, Solace has been very busy incorporating user suggestions into the product. At the top of the list of the few minor improvements I'd still like to see in VariView is for it to remember the column sizing on its browses. As it stands, the browse formatting goes back to the defaults each time the viewer window is closed and re-opened. Since I have a preference for using descriptive (i.e., long) variable names, I always had to resize some of the columns each time I popped up the VariView window.

Another suggestion would be to have a simple switch to disable all of the VariView code from the global extension. That would allow the developer to easily generate a version of an application that doesn't contain VariView code without disturbing the settings for each individual VariView procedure extension. As it currently stands, you would either need to set the 'Do not generate code for this procedure' flag on every procedure extension or completely remove the templates.

Summary

VariView works, and it works well. It provides you with a

tremendous amount of information for very little work, and fills the middle ground between messages and the Clarion debugger quite nicely. Unlike the MESSAGE() function, VariView has little or no impact on your program while it is running. And unlike the debugger, you can ship VariView as part of your application to assist with support in the field.

One thing to keep in mind is that VariView is only useful if your program is actually working. If you are trying to debug program crashes, in most cases you'll still need to fall back on the debugger or using MESSAGE () statements to pinpoint the exact source of your problems. For data-related problems, though, VariView is wonderful at letting you track through what's really happening inside of your files buffers and variables. Even better, you can view that information without disrupting program flow or adding and removing tons of messages (and without having to recompile at each step).

Bottom line: VariView is a very cool tool! I highly recommend that you check it out.

Overall Product Rating:	
Ability to do the task	
Ease of use	
Ease of Installation	
Documentation	
Technical Support	
Black-Box DLLs/LIBs	No

VariView 2.0e is compatible with Clarion 5 and 5.5, both ABC and Legacy template chains. It can be purchased online through [DeveloperPLUS](#) for \$US149.

For more information or to download a demo of VariView, please visit the Solace Software web site at: <http://www.solace-software.demon.co.uk>

Vendor Comments from Solace Software

We have take on board what the reviewer said about the installation and will be adopting most of the [C3PA](#) installation standards for the next release. This will include automatically bringing up the help file and registering the templates at the end of the installation.

We have also addressed the problem the reviewer had with 'Type' declarations and these will be automatically ignored in the next release. As they are not 'physical' entities, they cannot actually hold a value and therefore cannot be displayed with a value.

On the documentation side we have added the contact details as suggested. They are also on the template global extensions, and have now been added them to the help file as well.

The next release will include the requests for saving the Window/listbox settings between sessions and disabling all from the global extension.

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

Reader Comments

[Add a comment](#)

Copyright © 1999-2001 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca

Reborn Free

CLARION
*online*published by
CoveComm Inc.

Clarion MAGAZINE

Clarion and Multi-Edit: Together At Last

by **Vince Du Beau**

Published 2001-04-24

The Clarion product has always had a decent editor built into the IDE. While I never disliked the Clarion editor, I've used American Cybernetics Multi-Edit since it's early days of DOS and before I discovered CPD 2.1

The Windows versions of Multi-Edit can be integrated into a variety of IDE's such as Borland's Delphi and C++. The latest Multi-Edit beta adds integration with Visual Studio and Visual Basic. I decided that Clarion developers shouldn't be left out in the cold, so in this article I will detail a simple method of integrating Multi-Edit into the Clarion IDE.

Customizing the Clarion IDE

Unlike Clarion, the Borland products have a very sophisticated open interface to their IDE. This makes it possible for American Cybernetics to create a custom DLL and provide a very tight integration between the two products. The Clarion IDE does not provide this type of interface but does give us a simple way to change our via INI files.

One thing you can do with INI files is add items to the menu. (For a complete text on customizing the INI file see [Customizing Clarion5's Editor and Menus](#) by John Morter.) The name of the required INI file will vary depending on the Clarion version you are using. Since I am using C55EE, my INI file is C55EE.INI. To add Multi-Edit to the IDE menus, make the following changes to the INI file (after you make a backup copy).

```
[User Menus]
```

```
_version=41 1=Data Modeller|DIC5 2=&Tools/&MultiEdit|MultiEdit [User  
Applications] _version=41 CWRW=c5rw %f %a DIC5=DM5.EXE  
MultiEdit=c:\develo~1\multi~1\mew32.exe %f
```

The lines in boldface are the ones that need to be added. Figure 1 shows what this should look like when you run Clarion.

[Search](#)

[Home](#)

[COL Archives](#)

[Subscribe](#)

[New Subs](#)

[Renewals](#)

[Info](#)

[Log In](#)

[FAQ](#)

[Privacy Policy](#)

[Contact Us](#)

[Downloads](#)

[PDFs](#)

[Freebies](#)

[Open Source](#)

[Site Index](#)

[Call for](#)

[Articles](#)

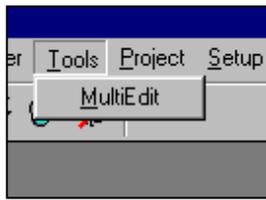


Figure 1. Adding Multi-Edit to the Tools menu.

Customizing Multi-Edit

Configuring Multi-Edit is not quite as simple, but it's not a daunting task, and can be done entirely from the Customize dialog in Multi-Edit. Choose Tools->Customize; Figure 2 shows the Customize dialog.

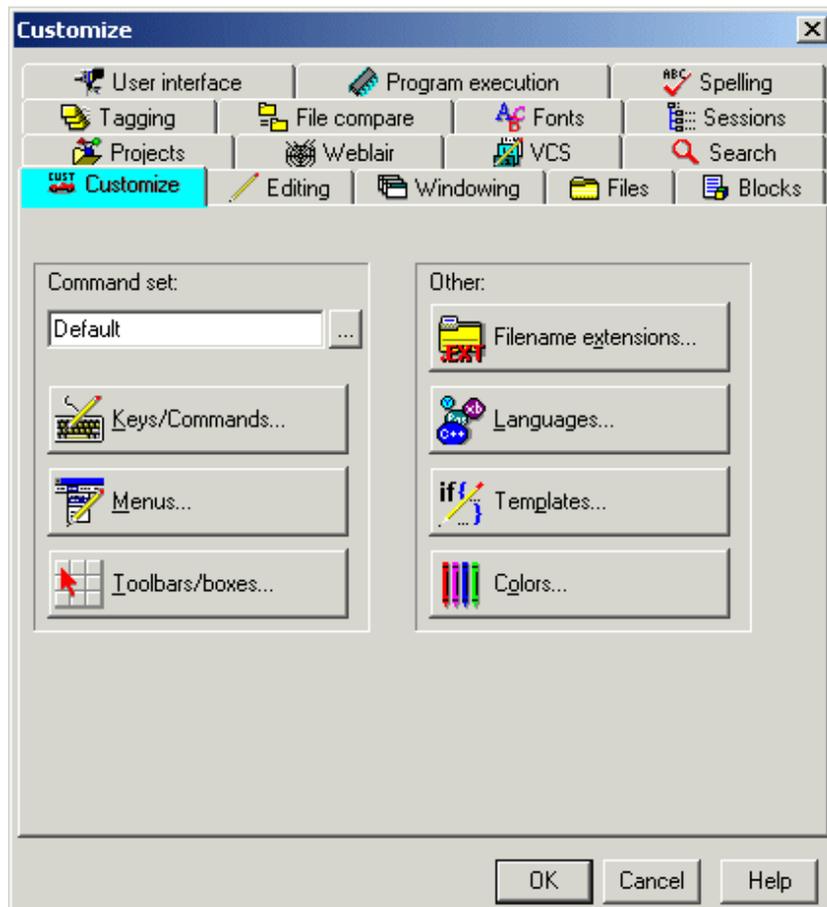


Figure 2. Customizing Multi-Edit

The first step is to add Clarion to the menu. Click on the Menus button to bring up the menu editor. Select main and click on Edit. The Edit Menu Set "Main" dialog will appear. Scroll down to the Help menu option, highlight it and click on the Insert button. This will bring up the next dialog, shown in Figure 3.

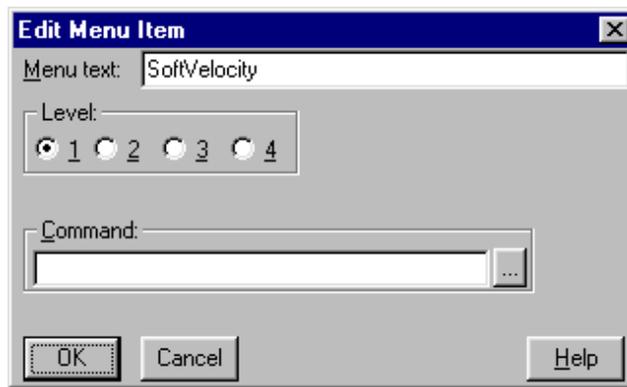


Figure 3. Adding a Multi-Edit menu item

Fill in the dialog as shown and click OK. Highlight the Help option and click Insert again. In the menu text type "Run Clarion" and check the 2 under level. Click on the ellipsis next to the command prompt. This will bring up the Command Mapping. You will have to add a new command, so click Insert and fill out the dialog as shown in Figure 4. If you are not using C55EE, substitute the appropriate executable. You can also change the command name and primary key if desired.

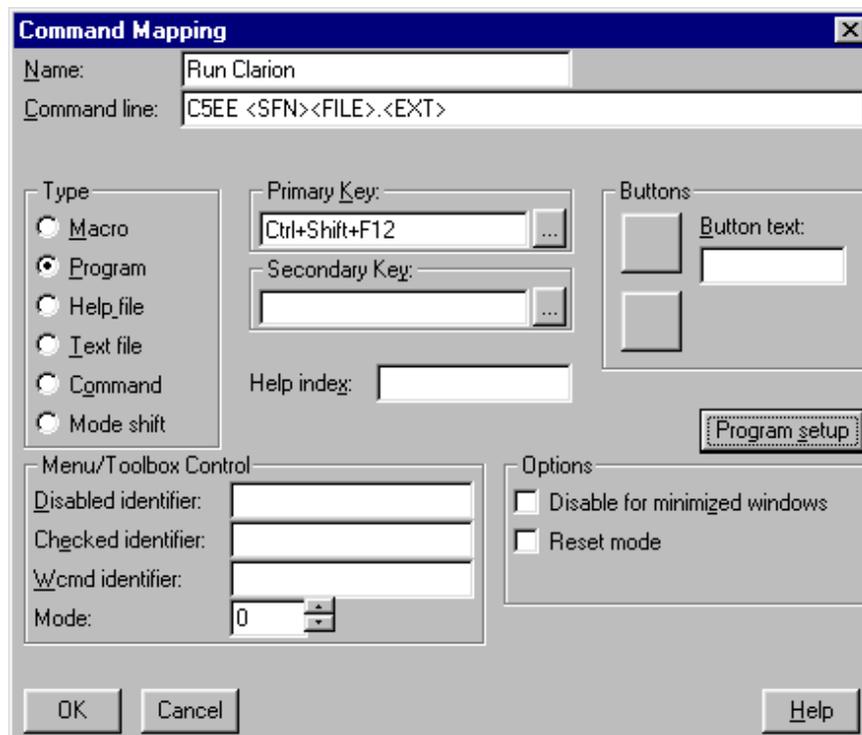


Figure 4. Creating a command mapping

The command as seen assumes your path includes the Clarion\bin directory. If not, you must supply the entire path name. The breakdown of this command line is as follows:

C5EE - The Clarion executable.

<SFN> - Instructs Multi-Edit to use short file names (until the Clarion IDE is 32-Bit).

<FILE> - The current file name minus the extension.

<EXT> - The current file's extension.

If you were editing MYPROG.CLW the command line would become C5EE MYPROG.CLW.

The <SFN> value is not passed to Clarion.

The Program Setup button is optional for this part of the setup. If you leave this as is, when you call Clarion, Multi-Edit will wait until Clarion closes before allowing you to continue. This isn't a problem but it was one of those little things that annoyed me. You can change this behavior by choosing program setup and then setting the Run In Background checkbox and unsetting the Wait For Program To Finish checkbox.

Click OK and you will be back to the Command Mapping dialog and your new "Run Clarion" command should be highlighted. Click Select, OK, Close, Close and OK. Your new menu option is set.

Syntax Highlighting

Of course editing your Clarion source wouldn't be the same without having your choice of colors for your code. Here's how it's done. Select the Languages button and then the Insert. Complete the dialog box as shown in Figure 5.

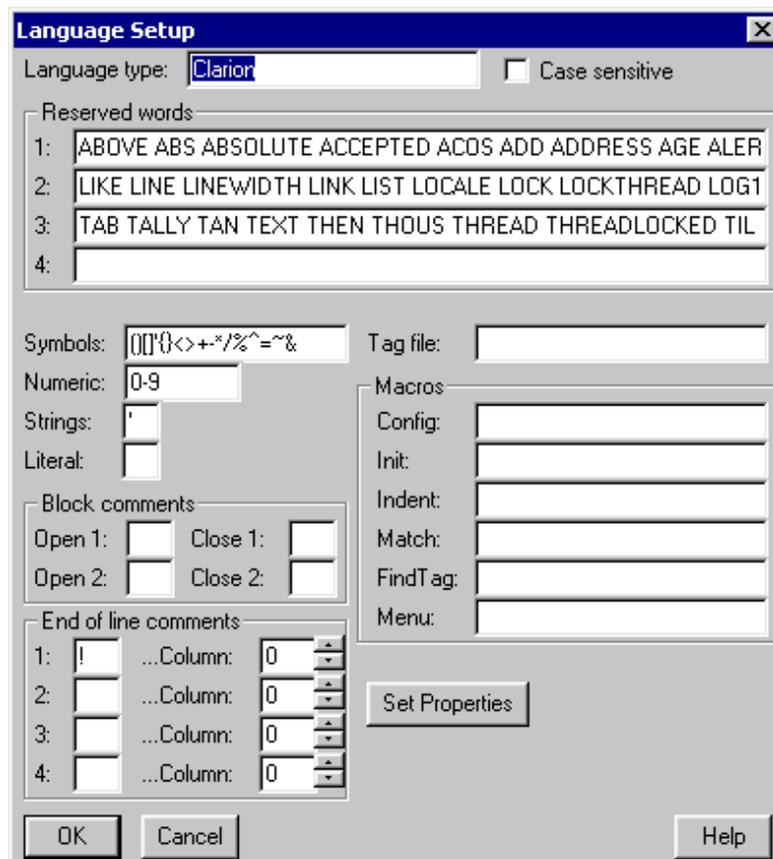


Figure 5. Specifying Clarion reserved words

To make your life easier, I've included a file called [ClarionKeywords.txt](#) which includes all the keywords for both the language and template language. Make sure that you open the file with Wordpad as the lines are over 1400 characters and Notepad does some strange things with it. There are three lines in the file. Just cut and paste them into each of the three fields show above. Click OK and Close and proceed to the File extensions button.

Not much has to be done here. In the extensions field type CLW, INC, TRN, TPW, TPL separated by spaces. Click the ellipsis for language and highlight and select Clarion. The default tab is set for 8. You should change this to your preference. The final step for the Clarion language is to click the ellipsis next to the colors field which brings up the syntax color setup box in Figure 6.

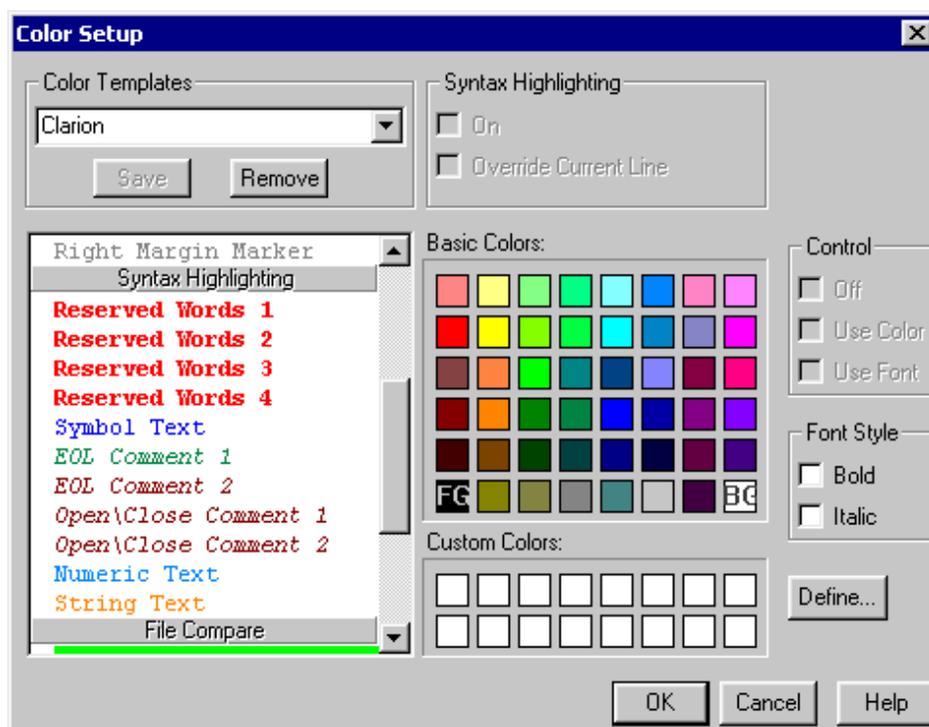


Figure 6. Setting the syntax highlighting colors.

Scroll down to the Syntax Highlighting section. Here you can pick the colors for the various source elements. You can have different colors for each of the three lines of reserved words you entered in the Language Setup dialog if you are so inclined. Make sure when you are done picking your colors to enter Clarion into the Color Templates field and hit the Save button. Hit OK and OK again. Multi-Edit show a message asking you to confirm "Reset extension configuration in currently loaded files?" If you have Clarion source code loaded into Multi-Edit click Yes, otherwise click No. Select the close button.

And To Finish

To complete the setup you should make sure the Autosave options

in Multi-Edit are set to ensure that the file is automatically saved and reloaded when you switch back and forth. This is done by selecting the Files Tab and then the Autosave Tab. Your settings should look like the ones in Figure 7.

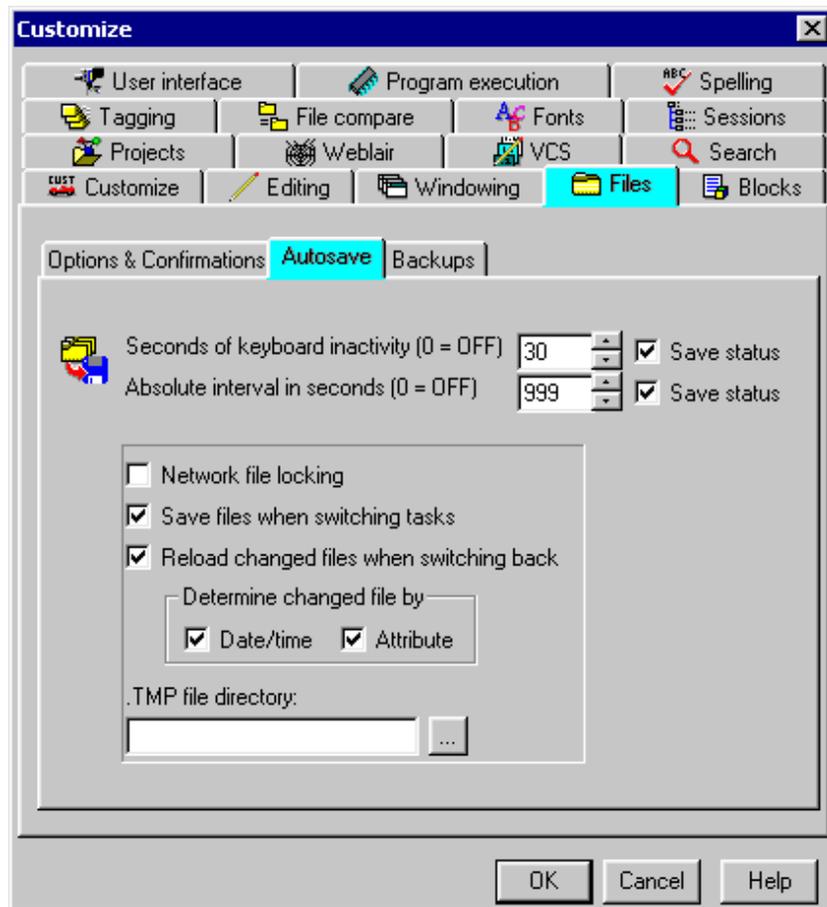


Figure 7. Setting the Autosave options

The final step to integration is to make sure that the Clarion editor will detect and reload the file if you change it in Multi-Edit. To do this you need to go into the Clarion Setup->Editor Options menu and selecting the Saving Tab. Make sure that the Prompt for Reload if file changed box is checked.

You can now call Clarion from Multi-Edit by pressing either the primary key that you assigned (Ctrl+Shift+F12 in the example) or by using the new menu option which should like Figure 8.

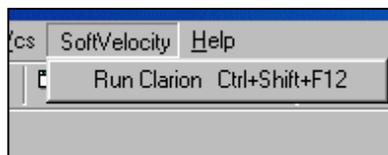


Figure 8. The Run Clarion option on the SoftVelocity menu

Summary

There are some drawbacks to using Multi-Edit as your editor. The

most obvious is not being able to call up the screen and report formatters with a simple Ctrl-F from within Multi-Edit. A minor issue will also arise if you also code in Pascal or Delphi. Since both Clarion and Pascal have .INC as a file extension, Multi-Edit will use the syntax highlighting for whichever language is in the list first.

One addition I would like to see in the Clarion IDE is expansion macros for the line and column numbers. Multi-Edit has the ability to both pass and receive these, so in theory the you could at least start at the same point in the code. You also have to stick with the Embeditor, since there is no way to customize its menus. But overall, for the diehard fans of Multi-Edit who have come to appreciate all of it's features, the drawbacks should be a minor problem.

[Download ClarionKeywords.txt](#)

[Vince Du Beau](#) is the host of the radio talk show talk Bit 'n Bytes on WALE from Providence, Rhode Island. His company, Plover Development Group Inc., does AS/400 consulting and custom PC development with Clarion.

Reader Comments

[Add a comment](#)

So has anyone tried this yet? I don't use Multi-Edit, but...

Hmm. Might make a good survey question...

I use it all the time, but obviously I'm a bit biased...

Obviously

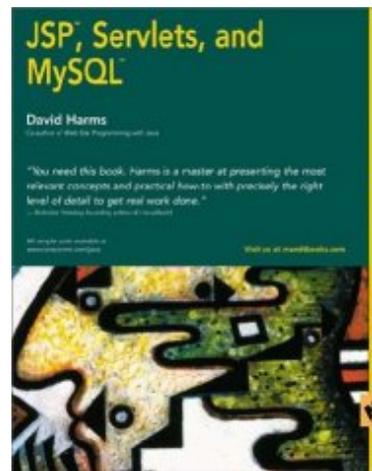
Copyright © 1999-2001 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca

Clarion MAGAZINE

No Mag This Week; Instead, A Book**by Dave Harms**

Published 2001-04-17

I had originally planned to publish several articles this week, as usual, but instead I've been busy preparing a [support web site](#) for my new book, titled [JSP, Servlets and MySQL](#) (published by M&T Books, an imprint of Hungry Minds, Inc. formerly known as IDG). I'm pleased to announce that the web site is now up, and the book is in the stores. Well, some stores – Amazon and Barnes & Noble both indicate they have the book in stock. It may take a little longer to arrive at your local bookseller.



What does all this have to do with Clarion Magazine, aside from interfering with the normal publication schedule? For starters, most of the code I describe in the book (which is all about building web applications with Java and MySQL) is actually in use, delivering Clarion Magazine.

For instance, every request you make for a page (well, almost every request) is handed off to a Java servlet, which is a Java class designed specifically to respond to HTTP requests. The servlet doesn't have a huge amount of intelligence – it mostly makes use of a number of JavaBeans, which handle functions like logging the requests to the database, looking up the requested article, verifying you have rights, and so forth. When all this processing is complete, the servlet hands the request off to a JavaServer Page (JSP), which is a little bit like an ASP page. You may think you're looking at an HTML page, but in fact you're looking at a JSP that acts as a wrapper for the document you've requested.

The forms handling code, the weekly survey, all of these are written in Java. And if that strikes you as just a bit traitorous, don't worry. Clarion is still a crucial part of the operation of Clarion Magazine. The web application that delivers Clarion Magazine depends on data in a MySQL database, and I know of no better

[Search](#)[Home](#)[COL Archives](#)[Subscribe](#)[New Subs](#)[Renewals](#)[Info](#)[Log In](#)[FAQ](#)[Privacy Policy](#)[Contact Us](#)[Downloads](#)[PDFs](#)[Freebies](#)[Open Source](#)[Site Index](#)[Call for](#)[Articles](#)

way to manage that data than with a Clarion application.

So if you're interested in Java web application development, curious about the inner workings of Clarion Magazine, or not sure which randomly selected computer book to buy next, may I recommend [JSP, Servlets, and MySQL](#). ISBN 0-7645-4787-9, 500pp.

*[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](#).*

Reader Comments

[Add a comment](#)

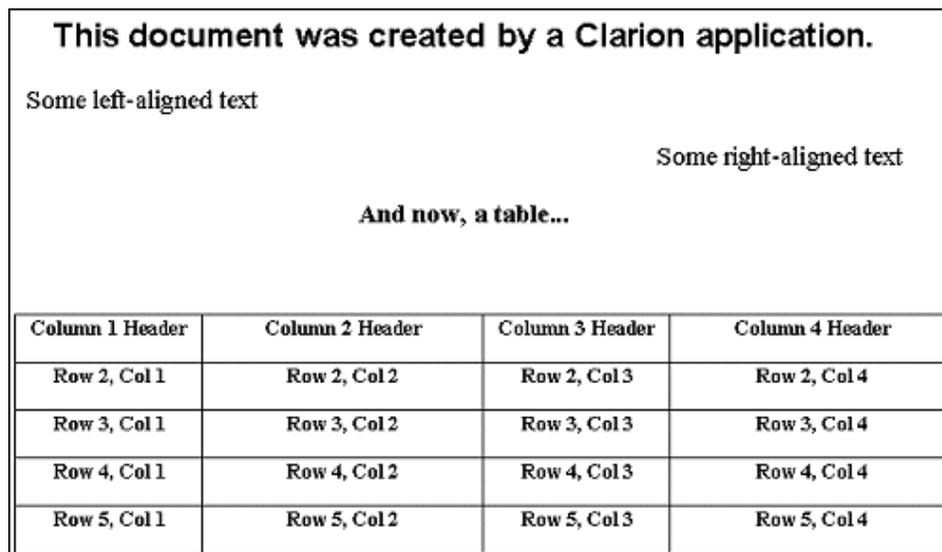
Copyright © 1999-2001 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca

Clarion MAGAZINE

Writing Classes That Create Word Documents**by Dave Harms**

Published 2001-04-10

This week I decided to employ some old technology to create Microsoft Word documents with Clarion. A while ago Andrew Guidroz sent me some code he'd been playing with that controlled Word using Jim Kane's OLE support code from November of 1999. Now Jim's latest OLE code would no doubt lend itself even better to this task, but I didn't have a ready example of what I wanted to accomplish using that code. Instead I took a page from the Lazy Programmer's Society handbook and adapted the example I had in hand. Figure 1 shows the result.

**Figure 1. A Word document created by a Clarion application**

Jim's code is really quite easy to use – you only need two files, oletcl.inc and oletcl.clw. Then you use some of his code to wrap your own calls to the OLE object. Here's the setup code, which gets a Word OLE object and creates a document:

```
oletcl.init('word.application')
oletcl.getobj('application',capplication)
oletcl.callmethod('documents.add')
oletcl.callmethod('documents(1).activate')
oletcl.getobj('activedocument', cactivedocument)
```

[Search](#)[Home](#)[COL Archives](#)[Subscribe](#)[New Subs](#)[Renewals](#)[Info](#)[Log In](#)[FAQ](#)[Privacy Policy](#)[Contact Us](#)[Downloads](#)[PDFs](#)[Freebies](#)[Open Source](#)[Site Index](#)[Call for](#)[Articles](#)

And here's the closedown code, which releases the document:

```
oletcl.releaseobj(cactivedocument)
oletcl.setprop('visible', '1')
oletcl.releaseobj(capplication)
oletcl.kill()
```

In between those two blocks of code you can call OLE methods to do a variety of tasks, including setting page margins...

```
oletcl.setprop(|
'activedocument.pagesetup.leftmargin', '72')
```

...specifying fonts...

```
oletcl.setprop(|
'activedocument.range.font.name', 'MS Sans Serif')
```

```
oletcl.setprop(|
'activedocument.range.font.size', '18')
```

...setting paragraph alignment...

```
oletcl.getobj('selection', cselection)
oletcl.setprop(cselection & |
'.ParagraphFormat.Alignment', '1')
```

...typing text...

```
oletcl.callobj(cselection , |
'TypeText("This document was created ' & |
'by a Clarion application."))')
```

...creating blank lines...

```
oletcl.callobj(cselection , 'TypeParagraph')
```

...and doing more advanced tasks like creating a table...

```
Loc:CurrentParagraphString = |
  oletcl.callobj(cactivedocument, 'paragraphs.count')
Loc:CurrentParagraph = Loc:CurrentParagraphString
oletcl.getobj('activedocument.paragraphs(' & |
  loc:currentparagraph & ')') & '.range', crange)
oletcl.callmethod('activedocument.paragraphs(' |
  & loc:currentparagraph & ')') & '.range.select')
oletcl.getobj('selection', cselection)
oletcl.callobj(cselection , 'TypeParagraph')
oletcl.Getobj(|
  'ActiveDocument.Tables.Add( |
    ' & crange & ',5,4)', ctable)
oletcl.setprop(|
  'ActiveDocument.Tables(1).Borders.Enable', '1')
oletcl.setprop(|
  'activedocument.Tables(1).Columns(1).Width', '100' )
oletcl.setprop(|
```

```

'activedocument.Tables(1).Columns(2).Width','150' )
oletcl.setprop(|
'activedocument.Tables(1).Columns(3).Width','100' )
oletcl.setprop(|
'activedocument.Tables(1).Columns(4).Width','150' )

```

Writing this kind of OLE code can become tedious, and by the time I got to where I was writing data into table cells I decided to create a small utility class to do the job for me. Here's a type definition for that class, along with a class instance:

```

TableClass      class,type
TableNumber     short,protected
oletCl          &oleTclType,protected
init            procedure(short TableNumberolet, ←
                oleTclType oletCl)
writeCell       procedure(short row,short col,string text)
                end

table           TableClass

```

The class methods are declared at the end of the procedure, as follows:

```

TableClass.init procedure(short TableNumber, ←
                        oleTclType ole)
    code
    self.TableNumber = TableNumber
    self.oletCl &= ole

TableClass.writeCell procedure(short row, ←
                              short col,string text)
csel  cstring(20)
    code
    self.oletcl.callmethod('activedocument.Tables(' |
        & self.TableNumber |
        & ').Cell(' & row & ',' & col & ').Select')
    self.oletcl.getobj('selection', csel)
    self.oletcl.callobj(csel , 'TypeText("'" |
        & clip(text) & '"')')
    self.oletcl.releaseobj(csel)

```

As you can see, the `writeCell` method does all the hard work of locating the table cell, writing the text, and releasing the selection object. With that method in hand, I can easily write text to individual cells:

```

table.Init(1,oletcl)
loop y = 1 to 4
    table.WriteCell(1,y,'Column ' & y & ' Header')
    loop x = 2 to 5
        table.WriteCell(x,y,'Row ' & x & ', Col ' & y)
    end
end

```

You can [download](#) the source code from the end of this article. If you want to play with this code, I have two suggestions. One is to build on

the example of `TableClass`, and write a class that provides a simplified wrapper to the OLE Word control.

Once you your wrapper class in hand, you may want to consider deriving another class that uses the new COM Interface code Jim Kane described in his more recent articles. You could, of course, write one class to handle the "old" OLE technique, and a completely new class to handle the "new" OLE technique. But that would probably waste a lot of code, and more to the point, would make it more difficult to plug the new class into your existing code. If you want to add new functionality to a class, just derive a new class. For instance, to completely change the functionality of my `writeCell` method, I create a new class based on the old one, and I redeclare `writeCell`:

```
TableClassB    class(TableClass),type
writeCell     procedure(short row,short col,←
              string text)
              end
```

I can then create a new method declaration for `writeCell`, while all of the other behavior of `TableClassB` is inherited from `TableClass`. And to use this class, I just change the `TableClass` instance declaration from:

```
table TableClass
```

to:

```
table TableClassB
```

I don't have to change any of my other code, because my code references only methods already declared in the base class; I'm not using any new methods.

Another way to approach this is to use a base class reference, and upcast the derived class (that is, treat it as a class further up the hierarchy). In this case the `TableClass` instance declaration is of a reference variable:

```
table &TableClass
```

To use a reference variable like this, I would normally create a new instance of the class:

```
table &= new TableClass
```

Instead, I want to upcast a derived class to `TableClass`:

```
table &= new TableClassB
```

There *is* a type mismatch between the reference variable type (`TableClass`) and the object created (`TableClassB`). The compiler doesn't complain, because `TableClassB` is derived from `TableClass`. The only limitation is that you can't call any methods which are *only* declared in `TableClassB`. What you're really doing is making `TableClassB` pretend to be an instance of `TableClass`.

If you create a base class that has all the methods you need, and you declare derived classes that implement specific kinds of functionality, you can easily switch from one class implementation to another with just one line of code (plus the class declaration). Apply this technique to your OLE wrappers (and other classes) and you'll be well-positioned to take advantage of new technologies as they come along.

[Download the Clarion 5.5 example application](#)

[Download the Clarion 5.0 example application](#)

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](#).

Reader Comments

[Add a comment](#)

Copyright © 1999-2001 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca

Clarion MAGAZINE

SetFilter to the Max

by **Steven Parker**

Published 2001-04-10

In the last [action packed episode](#) I examined the use of SetFilter in some depth. I claimed that the SetFilter method is not only the most common way of filtering in a browse, process, or report, it is almost always the most logically effective way to accomplish this filtering. Now it's time to apply that knowledge and use SetFilter directly.

As the LRM states, SetFilter "specifies a filter for the active sort order" and takes a "string constant, variable, EQUATE, or expression that contains a FILTER expression." The most important fact in this statement, as far as I am concerned, is that SetFilter takes a string (constant or variable) parameter.

The inference I draw from the documentation is that all I need to do is format a string that contains what I might have typed in a Record Filter prompt, pass it to SetFilter, and I'm in business. In other words, after checking a few template generated SetFilter statements, I can apply everything I know about variable and string manipulation.

I demonstrated this by examining a moderately complex filter, as shown in Figure 1.

Receiving Report

Date Received: 9/23/00

Sort by: Part Number

Part Number
PLU

One PO per Page?

Print Grand Totals?

Cancel Start

Figure 1. Select filter

Hidden under the drop down is a field to get a range of values. Once this window is completed, the I embed the following code:

```
! date filter
FilterString = 'PO:Date = LOC:Date'
!Field to filter on
Case LOC:SortBy
Of 'PLU'
  FilterString = Clip(FilterString) |
    & ' and (PO:PLU <= ' |
    & 'LOC:Datum and PO:PLU <= LOC:Datum2)'
```

```
Of 'Part Number'
  FilterString = Clip(FilterString) |
    & ' and (PO:Part <= ' |
    & 'LOC:Datum and PO:Part <= LOC:Datum2)'
```

```
End
ThisReport.SetFilter(FilterString)
```

Bingo, an effective filter.

But what about genuinely complicated filters? What about filters so case-riddled or with so many choices that they require an entirely separate window, a criterion window, to collect the user's selections?

Very complex filters

Consider, for example, the window in Figure 2, which has 11 variables. The user may select any number of variables among the 11.

Figure 2. Criterion window to collect user choices

The user must choose at least one indicator, as the following code demonstrates:

```

If ~ss_s and ~lcs_s and ~do_s and ~ge_s and |
  ~al_s and ~tqa_s and ~rr_s and ~fs_s and |
  ~mf_s and ~nu_s and ~acc_s
  Message('A minimum of one indicator ' & |
    must be selected',|
    'Please Select',Icon:Asterisk)
  Select(1)
  Cycle
End

```

That's well and good, but am I then to check each and every variable to determine whether or not to create a string element? Each of these variables corresponds to a file variable, so I end up needing to do something like:

```

If ss_s
  FilterString = Clip(FilterString) |
    & 'FIL:ss_s = ss_s '
End

```

and repeating this for *each* variable. Except ... it won't work, because I haven't included any logical (AND, OR, etc) operators. Without these I would end up with a string like this:

```
FIL:ss_s = ss_s FIL:nu_s = nu_s ...
```

This is not a valid expression; it's not even a syntactically valid Clarion statement. Therefore, it is quite unlikely to do what I want it to do.

What I really need is something like this:

```
If ss_s
  FilterString = Clip(FilterString) |
  & ' AND FIL:ss_s = ss_s'
Else
  FilterString = 'FIL:ss_s = ss_s'
End
```

Now I need to copy this block of code ten times and change the variable names (certainly I wouldn't re-type the entire block for each remaining variable!).

As inelegant as this *seems* to be, it has the virtue of working (which is, in my opinion, rather an important virtue). I can also check whether anything had been selected by using the following code:

```
If Len(Clip(FilterString)) = 0
  Message('A minimum of one indicator ' & |
  must be selected', |
  'Please Select', Icon:Asterisk)
  !etc
End
```

rather than checking each variable, as shown above. (Yes, checking the length of `FilterString` would be slower but I'm not really sure anyone would notice in a real-world application.)

If you think that was complicated...

Now consider the criteria selection window in Figure 3. It seems like the user can select up to 15 ranges in five categories to filter on. In fact, the way this window works, the user may complete only the "From" field, to specify a single value (a simple assignment), or both the "From" and "Through" elements, to specify a range of values.

Figure 3. Criterion selection window

Furthermore, the user does not have to start in the first field of a pair of entry fields. For example, in Figure 4, the user has selected a Vendor code of SV with Departments 10 through 47:

Figure 4. Sample user selection

In the case illustrated in Figure 4, my final filter should be:

```
INV:Vendor = 'SV' and (INV:Dept => 10 |
and INV:Dept <= 47)
```

Now, suppose the user had specified a second vendor, not in the "Through" field next to where "SV" had been entered, but in the "From" field below "SV." Now, my filter becomes:

```
(INV:Vendor = 'SV' |
  or INV:Vendor = 'thatVendor') and |
(INV:Dept => 10 and INV:Dept <= 47)
```

I think it is clear that the difficulty of forming the full filter expression raises exponentially with the number of entries the user makes.

To make matters a bit less messy, all of the variables on this window are local and there are only five of them. Each block of fields, Vendor, Department, User Sort, PLU and Part Number, is populated by a single dimensioned variable of six elements. I populated each block so that the odd numbered elements are on the left ("From") and the even numbered elements are on the right. This allows creation of additional entry fields without having to create additional variables. I simply change the DIM() attribute and recompile.

Vendor		
	From	Through
Select Vendors	1	2
	3	4
	5	6

Figure 5: Design of the window

The structure shown in Figure 5 also allows me to check for a "Through" entry without a matching "From" entry in a completely generic way. Where "Spot" is a local variable, set in the Accepted embed to equal the dimension, I use the following code:

```
If ~VendorSort[Spot - 1] and VendorSort[Spot]
  Message('You cannot enter an ' & |
    upper limit without a ' & |
    'lower limit.', 'Warning', Icon:Hand)
  VendorSort[Spot - 1] = VendorSort[Spot]
  Clear(VendorSort[Spot])
  Select(?-1)
End
```

This allows me to yell at the user and move an unpaired "Through" entry to its corresponding "From" field. I can also easily check whether the values are in the right order (this particular code is called only from even numbered fields):

```
If VendorSort[Spot] and |
  (VendorSort[Spot] < VendorSort[Spot-1])
  Message('Upper limit must be higher ' & |
    than lower limit.', |
    'Warning', Icon:Hand)
  Clear(VendorSort[Spot])
  Select(?)
End
```

The use of a local variable for the array index lets me write one block of code to handle all edits. But what is truly elegant is that, if I add more dimensions (i.e., more fields), I do not have to change this code at all. No matter how many more fields I add, not one keystroke of my verification code has to be updated because no subscript is referenced directly.

Creating the filter expression

The next step is to examine the user's entries and construct the filter expression. It struck me as easiest to handle each variable one at a time (essentially writing the code for one and copying it four times and changing the variable names). On further consideration, I only need to look at the first, third and fifth fields. If there is an entry in one of those fields and no entry in its mate (similar to the `Spot` logical, above), I have to do a simple assignment. If there is an entry in the mate field, I have a range.

Rather than walk you through the logic, I'll let you check the final code and my comments¹:

```
!check odd number fields only:
Loop i = 1 to 5 by 2
  !if there is a value:
  If VendorSort[i]
    j += 1
    !blank temporary holding string:
    LOC:TempFilter[j] = ''
    !if there is a right side ("mate") entry:
    If VendorSort[i+1]
      !set up range:
      Loc:TempFilter[j] = '(INV:Vendor => ' & |
        & AddQuotes2(VendorSort[i]) & ' |
        AND INV:Vendor <= ' & |
        AddQuotes(VendorSort[i+1]) & ')'
    Else
      ! otherwise setup simple assignment required
      Loc:TempFilter[j] = 'INV:Vendor = ' & |
        AddQuotes(VendorSort[i])
    End
  End
End
End
```

Next, I take the temporary strings, up to three, and create the expression for the current variable:

```
!initialize the string holding the final expression:
pVNDFilter = ''
!loop through the temp strings
Loop j = 1 to 3
  If j < 3
    !if there is another temp string:
    If LOC:TempFilter[j+1]
      !add the current temp string
```

```

        ! to the final expression with an "or":
        pVNDFilter = Clip(pVNDFilter) & ' ' |
        & Clip(LOC:TempFilter[j]) & ' OR '
    Else
        ! just add the temp string:
        pVNDFilter = Clip(pVNDFilter) & ' ' |
        & Clip(LOC:TempFilter[j])
    End
Else !j = 3 !last possible temp string
    pVNDFilter = Clip(pVNDFilter) & ' ' |
    & Clip(LOC:TempFilter[j])
End
End
Length += Len(Clip(pVNDFilter)) !length measure

```

Note: I check the length of the string to determine whether the user has made any selections. Also, the Else ($j = 3$) clause *is* necessary; when $j = 3$, the If LOC:TempFilter[j+1] clause will fail and the code will simply concatenate the current string. At the same time, j will go out of range and cause a very difficult to debug GPF.

The above is repeated for each variable, changing code as necessary, resulting in up to five working strings. I suppose I could create another loop to check and concatenate the temporary strings and, thus, create the full, final filter expression. But this seems like an awful lot of work, especially in light of the fact that SetFilter will do the concatenating for me.

The SetFilter method takes an optional second parameter which "uniquely identifies (and prioritizes) the filter so you can apply *multiple* filter conditions" (emphasis added). The documentation of this parameter continues, "If you set several expressions, each with a unique id, then *all* those expressions must evaluate to true to include an item in the result set" (emphasis added). This means that when there are multiple, prioritized identifying expressions, SetFilter concatenates them for me!

Further simplifying my life, when I call this procedure, I pass in five strings as externals:

```
SortRoutine(*Cstring,*Cstring,*Cstring, ←
    *Cstring,*Cstring),Byte
```

The five strings I formatted, above, are used directly by the calling procedure, which is the procedure that is to be filtered:

```

If SortRoutine(VNDFilter,DeptFilter,USRFilter, |
    PLUFilter,PartFilter)
    ThisProcess.SetFilter(PLUFilter,'p1')
    ThisProcess.SetFilter(VNDFilter,'p2')
    ThisProcess.SetFilter(DeptFilter,'p3')
    ThisProcess.SetFilter(USRFilter,'p4')

```

```
ThisProcess.SetFilter(PartFilter, 'p5')
```

Et voila! I'm done!

If the user made no selection for a particular variable, I don't have to concern myself. The View Engine handles it. Nothing like letting the library do some of the work for you, is there?

Summary

`SetFilter` *is* enormously powerful. And it gives me enormous flexibility. And, all I have to know to use it is how to format a string (and occasionally remembering to refresh the window).

I can live with that.

Notes

[1](#) This code is a modification of code posted in a newsgroup by Arnor Baldvinsson and later updated by Arnor to show me this technique for constructing complex strings in a simple loop.

[2](#) `AddQuotes` is a neat function written by Alexey Solovjev. It is designed to examine a parameter, determine whether it is a string or numeric and, if a string, return it with correct apostrophes for use in Clarion strings and expressions.

```
AddQuotes PROCEDURE (? V)

    CODE
    If IsString(V)
        Return '<39>' & Clip(V) & '<39>'
    Else
        Return Clip(V)
    End
```

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 competitors' right side mirrors - while on the track (but only while accelerating). Steve has been writing on Clarion since 1993.

Reader Comments

[Add a comment](#)

Copyright © 1999-2001 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca

Reborn Free**CLARION**
*online*published by
CoveComm Inc.

Clarion MAGAZINE

Clarion and the Internet: Publishing Static Data

by Tom Ruby

Published 2001-04-03

Publishing static data to a web page using Clarion couldn't be easier. I've been doing it for years. Since July of 1997, to be exact. You can see the results at <http://www.tomruby.com/clarion>. I use the same technique to make my photo pages. (Yes, Steve, I know I need to update those, but you talked me into writing this article so I haven't had the time.)

To publish data to a web page, you just use Clarion's handy HTML (Hypertext Markup Language) driver. "Clarion has an HTML driver?" you ask, reaching for the phone to call Softvelocity. Of course Clarion has an HTML driver! HTML is nothing but ASCII text. You can write HTML with Notepad. In fact, the index.htm you get when you go to the Clarion Connection was written with notepad. If you're a real programmer, you might prefer EDLIN, but I like UltraEdit, which was recommended to me by a certain data ferret.

Anatomy of HTML

An HTML page is an ASCII file divided into three sections: . The header, followed by the body, followed by the footer. I think of these sections as the stuff before the stuff, the stuff, and the stuff after the stuff.

The Header

The header section contains information about the document. This happens to be the header from my main home page:

```
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1">
<meta name="GENERATOR" content="NotePad -
HTML editor for Windows">
<title>TomRuby.com</title>
```

[Search](#)[Home](#)[COL Archives](#)[Subscribe](#)[New Subs](#)[Renewals](#)[Info](#)[Log In](#)[FAQ](#)[Privacy Policy](#)[Contact Us](#)[Downloads](#)[PDFs](#)[Freebies](#)[Open Source](#)[Site Index](#)[Call for](#)[Articles](#)

```
</head>
```

First, you see `<html>`. This tells the web browser how to read the rest of the file. HTML is part text the browser will show on the page, and part tags telling the browser how to display the text. These tags are enclosed in `<` and `>` characters. Some tags occur on their own, while other tags are paired.

The Header is all the stuff between the two paired tags, `<head>` and `</head>`, and most of what I've shown here is optional. I created this header by having an HTML editor make a page, then opening that page with Notepad and editing it to say what I want.

The Body

The body consists of everything between the `<body>` and `</body>` tags. This is what will show in the browser's window. The following is the body tag from my home page.

```
<body bgcolor="#FFFFFF" link="#FF00FF"
text="#000000">
```

You see that this body tag specifies a background color (bright white), the color to display a link, and the color to use for the text. Don't believe me? Point your browser to <http://www.tomruby.com/home.htm> and then view the source (on Internet Explorer choose View|Source, on Netscape choose View|Page Source).

The `<body>` tag also lets you specify a background image, but if you're not careful these can make your page difficult to read.

There are two more tags you'll need to know about. The `<P>` tag causes the browser to make a paragraph break, and the `
` tag forces the browser to start a new line. The browser ignores carriage returns and extra spaces in the document, and formats the lines to fit the window on the reader's screen. When you create HTML you don't get full control over the final display; instead you instruct the browser to build the display for you, and the actual appearance is up to the browser.

The Footer

The footer just closes the body and the document with two tags:

```
</BODY>
</HTML>
```

You can fire up Notepad or another editor and type the header and body tag, put some text and close it with the footer tags, save the document with an extension of `.htm`, and IE or Netscape will show you the formatted HTML.

Output with a Process

That's all there is to creating HTML. To generate HTML with Clarion, you just write this text using the ASCII driver. Take any old Clarion dictionary you have and add a file. I called mine `HTML`. Specify the ASCII driver, and give the file one big field. I called the field `Text` and made it a `STRING(20000)`.

Now, open or create an application that uses this dictionary, and create a process procedure on some table (other than your `HTML` table). It doesn't matter which table you use as long as it has some data because this is just a demonstration. Add the `HTML` file to the process procedure under Other Files.

Next, go to the embed tree and find the `Init` method under the `ThisWindow` object (it's under Local Objects). Open the code branch and find where it says Open Files. Add an embed right above that. If you gave the file a variable filename, here is where you set the variable to the name of the file. You'll also want to add `REMOVE(HTML)` to empty the file, or the process will just append any HTML the end of what it created the last time you ran the process.

Add another embed after the Open Files mark. Put the code to write the header and the start of the body here. It will look something like this (edit for your own purposes):

```
HTM:Text = '<<html>'
Access:HTML.Insert()
HTM:Text = '<<head>'
Access:HTML.Insert()
HTM:Text = '<<meta http-equiv="Content-Type" '
Access:HTML.Insert()
HTM:Text = ' content="text/html;
charset=iso-8859-1">'
Access:HTML.Insert()
HTM:Text = '<<meta name="GENERATOR"
content="NotePad - HTML editor for Windows">'
Access:HTML.Insert()
HTM:Text = '<<title>TomRuby.com<</title>'
Access:HTML.Insert()
HTM:Text = '<</head>'
Access:HTML.Insert()
HTM:Text = '<<body bgcolor="#FFFFFF"
link="#FF00FF" text="#000000">'
Access:HTML.Insert()
```

Don't forget that the `<` character in a string in the source code means something to Clarion! When Clarion sees a single `<`, it assumes that whatever follows will be a character code, such as `<20>` for a space character. Be sure to put `<<` so Clarion will know you want a `<` in the text. Notice that the `>` character won't cause a problem.

If you want to display anything before the actual records, add that code right after you create the body tag.

While you're in the embed tree, go to the Kill method and add an embed to write the footer just before the parent call. My code looks like this:

```
HTM:Text = '<</body>'
Access:HTML.Insert()
HTM:Text = '<</html>'
Access:HTML.Insert()
```

All that's left to do is write the records. Find the TakeRecord method under the ProcessManager in the embed tree and add the code to write the data there. Remember, the browser is going to completely ignore line breaks, so you don't have to worry about getting all the information in one insert. The browser will put a line break wherever you put the
 tag and a paragraph break wherever you put the <p> tag. Some consider it a good idea to enclose paragraphs in paired <p> and </p> tags. The code might look something like this:

```
HTM:Text = FORMAT(REC:Date,@D17)
Access:HTML.Insert()
HTM:Text = CLIP(REC:LastName) |
  & ' ' & CLIP(REC:FirstName) |
  & REC>Note
Access:HTML.Insert()
HTM:Text = '<<P>'
Access:HTML.Insert()
```

If you run this code, and have all the typos and goofs fixed, you should be able to open the HTML file the program has created and see your records in the browser. It's not very pretty, but hey, you're just learning how to do it. Once you understand how it all works, you can make it much better.

Transferring the file with FTP

All that remains is to get the HTML file to the web server where the world can see it. Most web hosts allow you to use ftp to maintain your web site, so you might as well use Window's built in FTP (File Transfer Protocol) program.

The Windows FTP program is a command line utility for manipulating files on any FTP server, including your web host. You can run this program from Clarion with the RUN command like this:

```
RUN( 'ftp -s:ftpscript.txt', 1 )
```

Make sure you put this RUN command after the file is closed. Perhaps a good place is in the kill method.

You can get some documentation on this ftp program by getting a

command prompt and typing `ftp -?` but you probably won't find the output very helpful. The main feature that's going to matter is scripting, which lets you feed a set of commands in a text file using the syntax:

```
ftp -s:filename.txt
```

Here's the script file I use to update the Clarion connection:

```
open www.tomruby.com
tomruby
*&(amp;*&%$
Ascii
cd clarion
put clarion.htm clarion.htm
put cats.htm cats.htm
binary
put linking.jpg linking.jpg
put linksimg.jpg linksimg.jpg
disconnect
quit
```

The first line tells the ftp program to open the FTP server. The next two lines are the user id and password for the FTP server. No, I'm not going to show you my password. The next line tells the ftp program to do an ASCII transfer of the HTML files. The `cd` command switches to the Clarion directory on the ftp server. Then you see two `put` commands. They specify the source file name and the name you want the file to have on the web server. After that, I switch to binary mode and upload the two linking jpg's that Vernon Godwin of [Kefren Designs, Inc.](#) made for me. How I change the date on the graphics is a good subject for another article.

The script closes with a `DISCONNECT` command which logs off the ftp server and a `QUIT` command which closes the ftp program. When I call the `RUN` command I specify a `waitflag` parameter of 1 so the Clarion program will wait for the ftp program to exit. Then I use the `PlaySound` API to play a wave file so I know the ftp is done.

Beautification

This HTML generation program works, but I have to agree, it's ugly. First I'll make the program easier to use, then I'll pretty up the HTML.

You've probably noticed that typing HTML text into embeds is a gross way to edit HTML. But wait – HTML is just ASCII text, which can be easily stored in a database. I made a table I called `SNIPS` because it holds snippets of HTML for the program to include in the output. The table has two columns, called `Name` and `Contents`. `Name` holds an identifying name or phrase, and `Contents` is a long `CSTRING` which holds the HTML. I made a simple browse and form to maintain the snips table. Remember, HTML is just text, so you

can type HTML into a text control.

The program that generates the Clarion Connection uses SNIPS for the header, body tags (there are two sets, one for the left pane and one for the right), the summary at the bottom, what to put before a link, what to put if the link has the C3PA box checked, and so on. Now, instead of building that long embed to build the stuff before the stuff, I just do something like this:

```
SNI:Name = 'Header'
Access:Snips.Fetch(SNI:NameKey)
HTM:Line = SNI:HTML
Access:HTML.Insert()
```

You can easily build this into a procedure so you can just use the code:

```
AddSnip('Header')
```

Whew, it's a lot nicer editing the HTML in text controls than the embed editor! And you don't have to recompile to make changes.

The next thing you'll probably want to do is get your data into nice columns, and for that you need to make an HTML table. A table starts with a <table> tag and ends with a </table> tag. A table can have any number of rows and cells. It can also have divisions, but this isn't an HTML article, it's a Clarion article!. In a simple table, you just list the columns, row by row, rather like this:

```
<table border="0" width="100%">
  <tr>
    <td> Row 1 Cell 1
    </td>
    <td> Row 1 Cell 2
    </td>
    <td> Row 1 Cell 3
    </td>
  </tr>
  <tr>
    <td> Row 2 Cell 1
    </td>
    <td> Row 2 Cell 2
    </td>
    <td> Row 2 Cell 3
    </td>
  </tr>
</table>
```

There are lots of options and parameters you can use on tables. For example the border="0" parameter gets rid of the bars between the cells. You want to know what all these options are? The easiest thing to do is fire up your favorite HTML editor, make a table like you want, and open the results with Notepad or another plain text editor.

To get your data into nice columns, you just need to write the Table tag in the embed where you wrote the header, and write the table row with all the cells in the take record method. Don't forget to add </table> in the footer, right before </body>.

One thing to be careful with using tables: Remember when the Clarion Connection's predecessor, the TopSpeed Turnpike, gave you the navigation panel on the left, then waited a couple of eternities and suddenly flashed all the information on the right side at once? No? You must have a fast Internet connection. I have a modem, and believe me, it was a long wait! Most web browsers don't show you the table until they know how wide the cells are going to be, and since the columns aren't specified at the top of the structure, the browser has to receive the whole table before it will show you any of it. The whole Turnpike was one big table with a narrow cell on the left with a yellow background, and a narrow cell on the right with a white background. Everything else was in the middle cell, including a few more tables. The solution is to keep your tables fairly small, and to use multiple tables instead of one big, complex table.

Spilled Beans

There you have it. I've spilled the beans on how the Clarion Connection works. Well, almost. Remember that HTML is just ASCII text. Each of the links in the database has a text box where I put the HTML code for the description. There I can use tags to make the lists, bold or italic type and whatever else I want and the process procedures just adds the HTML to the output just like from the snips table.

The Clarion Connection is actually made with three process procedures. One builds the navigation section on the left side. Another writes the start of the right side including the new links section, and the third writes the rest of the right side including the wrapup at the bottom. You can use a technique like this whenever the data is fairly static. That is, you "publish" the data once for many people to see. I used this same idea to make my photo pages at <http://www.tomruby.com/rubys.htm>. Each picture has an HTML page, but they're all the same save for the caption, title, picture and link. It got very tedious getting all these right, so I made a little database and Clarion does all the grunt work for me. I think if I was doing a knowledge base, instead of putting the Clarion program on the web, that I'd have a program to create the index page and write nicely formatted HTML articles from the database.

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.

Reader Comments

[Add a comment](#)

Copyright © 1999-2001 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca

Clarion MAGAZINE

Dynamic Filters: Applying The Theory

by **Steven Parker**

Published 2001-04-03

In the [last exciting episode](#), I asserted that there are three and only three things you need to know to effectively implement a filter:

- (1) the user interface is irrelevant
- (2) there are only two ways to effect a filter in ABC

and,

- (3) filters can include or exclude records, nothing else.

I stated that the interface is irrelevant because any Clarion developer who has completed the tutorials can create any user interface required.

My claim that there are only two ways to filter in ABC is based on the fact that I can either affect the view underlying a browse, process or report or I can examine each record as I loop through the file (again, in a browse, process, report or hand-coded procedure). There are *many* ways of implementing these two approaches, but there are no other strategic options.

My final claim, that filters can include or exclude records and nothing else, is most important when going through all records, in a loop (a process or report is, essentially, a loop, don't forget). By carefully considering the earliest point at which a fully determinative condition is fulfilled, you can optimize complex filters with a `Return Record:Ok` or `Return Record:Filtered`. I have [already covered this](#) in some depth, so I will not consider it further here.

Now, let's take a look at applying the theory and examine some filters, from the sublime to the ridiculous (and, I'll let you determine which is which).

The Method of Choice

[Search](#)

[Home](#)

[COL Archives](#)

[Subscribe](#)

[New Subs](#)

[Renewals](#)

[Info](#)

[Log In](#)

[FAQ](#)

[Privacy Policy](#)

[Contact Us](#)

[Downloads](#)

[PDFs](#)

[Freebies](#)

[Open Source](#)

[Site Index](#)

[Call for](#)

[Articles](#)

Most Clarion developers use the `SetFilter` method for filtering their browses, processes and reports, whether they know it or not. In fact, this is undoubtedly the most common and effective way to filter a file.

From the LRM, `SetFilter` "specifies a filter for the active sort order" and takes a "string constant, variable, EQUATE, or expression that contains a FILTER expression." The best part about the `SetFilter` method is that I don't even have to call it explicitly! I do not have to call it explicitly because entering an expression in the Record Filter prompt, as in Figure 1, actually creates a `SetFilter` call for me.

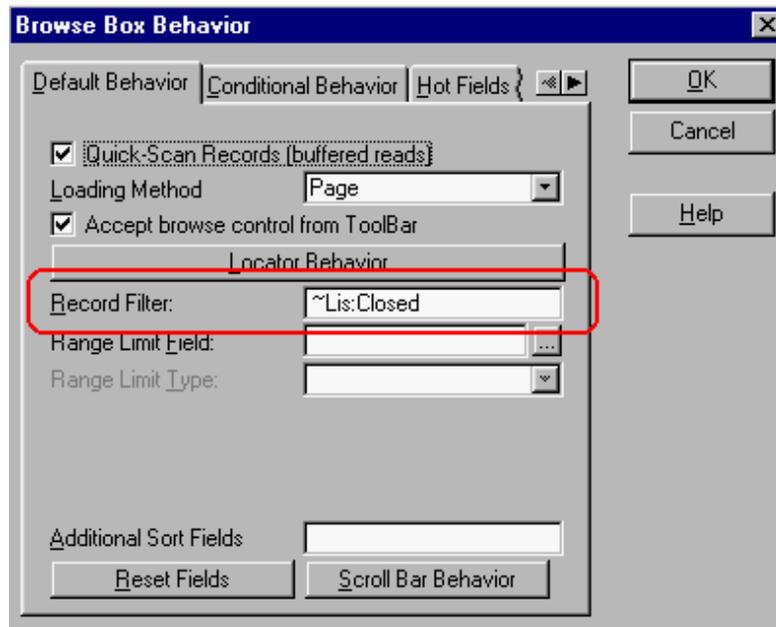


Figure 1. Using the Filter prompt.

The Filter prompt entry in Figure 1 generates the following code for into my source file:

```
BRW1.SetFilter('( ~LIS:Closed )')
```

In the case of a browse, with multiple tabs, all of which must use the same filter, it is necessary to enter or to copy the filter into the Record Filter prompt for each tab (choose Procedure Properties | Browse Box Behavior | Conditional Behavior and select the tab then click Properties).

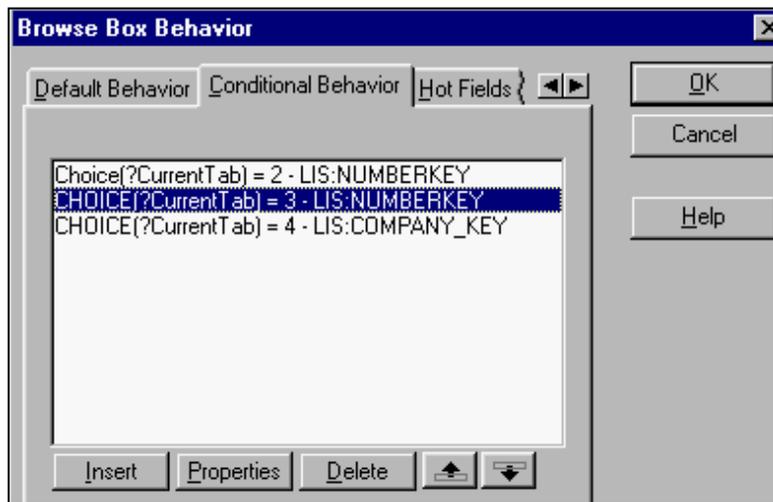


Figure 2. Using the Filter prompt with multiple tabs:

In the examples above, I simply check that LIS:Closed (a date) does not have a value. This is a static filter, but it is still instructive. It is also possible to use the Record Filter prompt with variables. These variables may be primed at any time. Prudence, of course, dictates that filter variables in reports and processes be primed before the report or process starts. But, in browses, I can make the variable available to the end user for entry, on the browse window, at runtime.

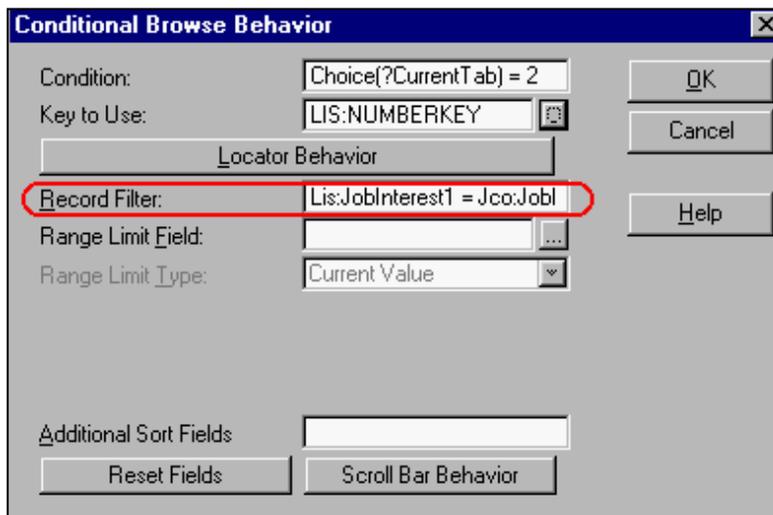


Figure 3. Filter prompt set up for runtime variables

In the example shown in Figure 3, LIS:JobInterest1 is a variable from the browsed file (i.e., it is a field in the view). JCO:JobInterest1 is a variable from another file, and I capture its value in a File Loaded Drop displayed on the currently selected tab.

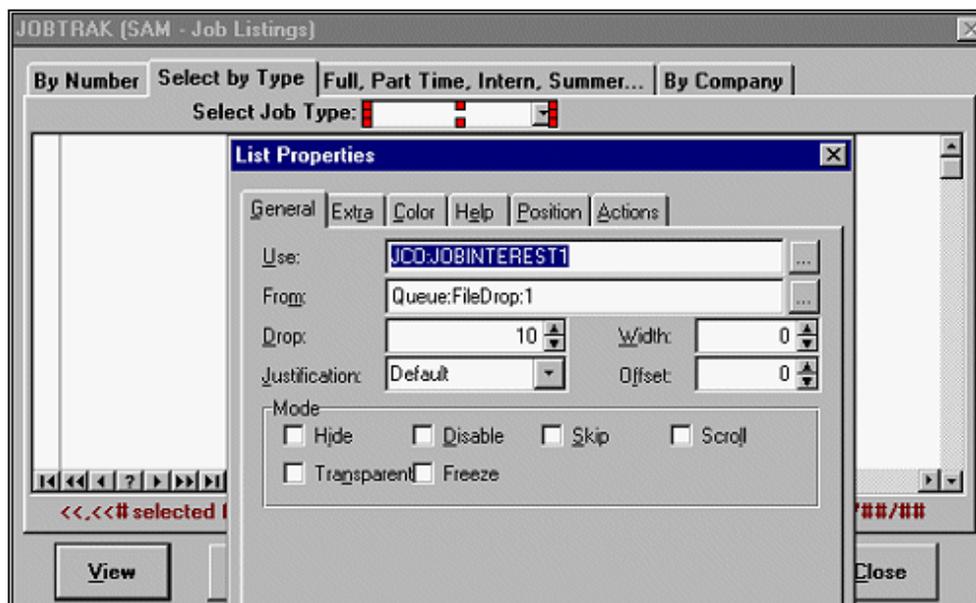


Figure 4. File Loaded Drop box for priming filter.

All that I have to do is refresh the window after the file drop is completed. `ThisWindow.Reset(1)` in the file drop's `Accepted` embed will do the job and the qualifying records, if any, will display.

Now I have a filter that is dynamic. When the user selects the tab with this file drop for the first time, `JCO:JobInterest1` has no value, so `LIS:JobInterest1 = JCO:JobInterest1` will be false and the browse box will be empty. Once the user makes a selection, the filter may be met and the browse will fill. (In the event that there are no matches, I do have code to display a "no records" message based on the example in "Making the Transition to ABC," on the CD.) Note that since the filter is specific to a particular tab, if the user returns to a tab the previous filter selection will still be in effect. The list is refreshed *only* when a selection is made in the file drop.

Entry fields, checkboxes and option boxes can be used in the same way. This kind of filtering is dynamic, and it's easy. The templates handle all the hard work for me. All I do is type in the desired filter and provide the required controls.

Filtering Manually with SetFilter

Using the Record Filter prompt makes filters, static or dynamic, very easy (unless you forget to `Bind` the variables or fail to include file variables in the view – i.e., make them hot – in which case, bad things begin to happen¹). But, more than that, examining the code generated by using that prompt teaches me just about all I need to know about using the `SetFilter` method in embedded hand code. And, there are a number of cases in which I find it expeditious to call `SetFilter` without the assistance of the templates.

One of the most obvious reasons to call `SetFilter` manually is to avoid having to complete numerous Record Filter prompts. If a browse has a few tabs, it may not be too inconvenient to copy and paste a filter expression from the main Browse Box Behavior worksheet. But if the browse has a large number of tabs, this may turn into a real ... impediment to ongoing felicity (but you know what I really meant, don't you?).

In this case, completing the Record Filter prompt on the main Browse Box Behavior worksheet (ensuring that the browse opens filtered²) and manually calling `SetFilter`, with the appropriate expression, can eliminate the need to enter and re-enter the filter. Copy the `SetFilter` statement created by the Record Filter prompt and paste it into the browse's `NewSelection` embed (before or after `Parent Call` doesn't matter).

Unfortunately, the user will have to tab back and forth a few times before the tabs all coordinate correctly. In the sample app accompanying this article, adjust the code in the `NewSelection` embed, by commenting/uncommenting lines like the window refresh or the `Post(Event:Accepted ...)`, and test. You'll see what I mean.

There are two solutions to coordinating the tabs. The most often recommended solution is to follow the `SetFilter` statement with `ThisWindow.Reset(1)`. And, if you do this, all tabs will display correctly.

However, when I examine the code generated by the templates when I complete the Record Filter prompt for more than one tab, I discover that the templates set the filter in the `ResetSort` method. Since the `ResetSort` method takes a "force" parameter, just like `ThisWindow.Reset`, calling `BRWx.ResetSort(1)` should also work. And indeed it does. This approach is also very effective in Internet Connect and Web Builder apps.

Which method call, `Reset` or `ResetSort`, is better? Simple. Whichever is faster with the particular filter being implemented.

If you are thinking that you can move the `SetFilter` call into the `ResetSort` method since, after all, that's what the templates would have done for you, you're right. However, the user will still have to tab around a bit to sync all the list boxes. On the other hand, if you have to go into the Conditional Behavior tab for each browse tab anyway, say to set up a locator, this technique may not gain you much.

Conditional Filters

The other obvious use of manually calling `SetFilter` is a conditional filter. For example:

```
If Auth_Level < 77
    !restrict customers shown
```

```

ThisReport.SetFilter('CUS:Bal <= 12')
Else
  ThisReport.SetFilter('') !no restriction
End

```

Possibly the most-liked application of manually calling `SetFilter` is the ability to offer the user a selection of filters at runtime.

For example, before a report or process will open and display the Progress window before beginning to process records. There is a template called `PauseButton` which allows me to stop the process immediately when the window displays (before any records are read). Using this template I can collect information from the user (and, because `ProgressWindow` is local to the procedure, I can use local variables, should I so desire).

I can offer a control that lets the user select to filter in various ways, as shown in Figure 5.

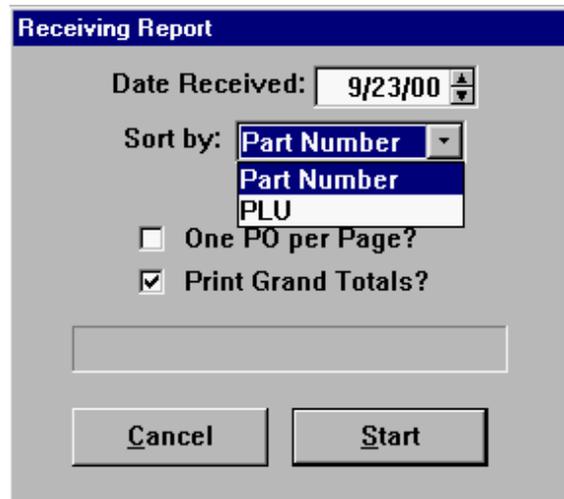


Figure 5. Letting the user select a filter.

(Hidden under the drop down is a field to get a range of values.)
Once this window is completed, execute the following code:

```

FilterString = 'PO:Date = LOC:Date'
Case LOC:SortBy
Of 'PLU'
  FilterString = Clip(FilterString) |
    & ' and (PO:PLU <= ' &|
    'LOC:Datum and PO:PLU <= LOC:Datum2)'
Of 'Part Number'
  FilterString = Clip(FilterString) |
    & ' and (PO:Part <= ' &|
    'LOC:Datum and PO:Part <= LOC:Datum2)'
End
ThisReport.SetFilter(FilterString)

```

Since `SetFilter` takes a string (constant or variable), all I need to do is format a string that contains what I might have typed in the

Record Filter prompt. In other words, I can apply everything I know about variable and string manipulation.

Clearly, this technique is easily extended.

Resetting the Filter at Runtime

SetFilter is also the method of choice when I want to set and *unset* a filter at runtime.

Consider a product browse like the one in Figure 6. I may want to display in-stock items only or I may want to see all items (most shopping cart apps offer such a feature).

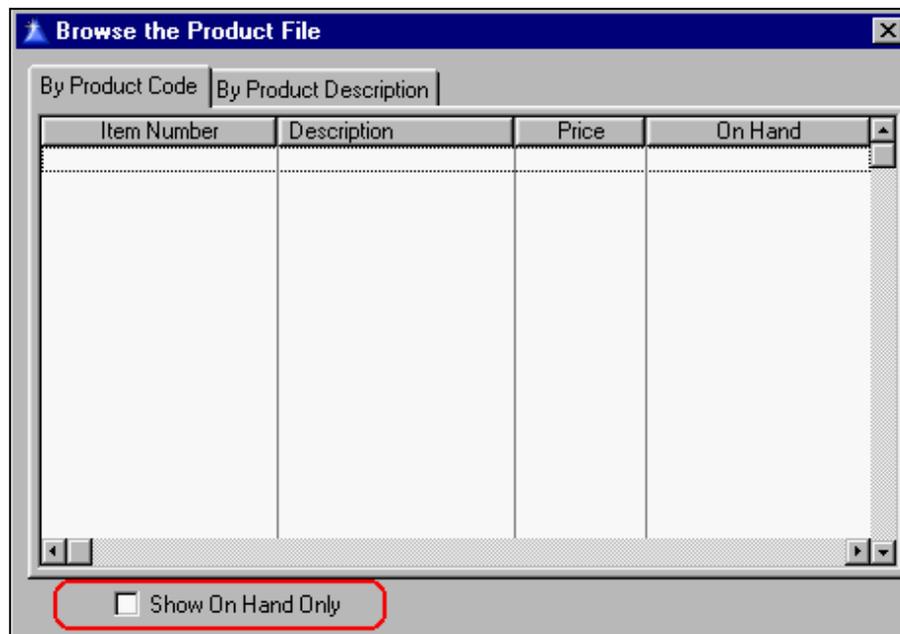


Figure 6. A Product browse

If the checkbox is ticked, I want:

```
BRW1.SetFilter('PRO:OnHand')
```

But if it is not checked, I want no filter:

```
BRW1.SetFilter('')
```

So, when the checkbox is Accepted, I want to re-set the filter like this:

```
If LOC:Filter_
  BRW1.SetFilter('PRO:OnHand')
Else
  BRW1.SetFilter('')
End
BRW1.ResetSort(1) !or ThisWindow.Reset(1)
```

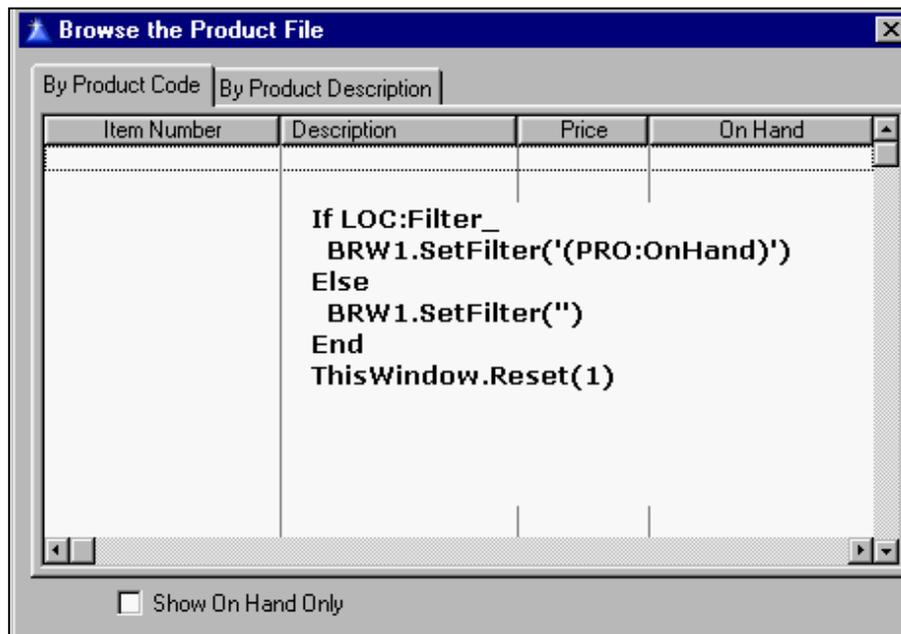


Figure 6. SetFilter for Runtime Set/Reset

Finally, to ensure that the state of the checkbox is current when changing tabs, add the following in the sheet's NewSelection embed:

```
Post(Event:Accepted,?LOC:Filter_)
```

(This has the same effect as putting the filtering code, above, into a local method or a routine and calling it from both the control's Accepted embed and from NewSelection, thus avoiding writing the filtering code twice.)

By extension, I can use SetFilter on browses without tabs and, therefore, on any window used for gathering user input.

Suppose I have a variable, FilterString, and a control to capture a value for it. This control can be an entry field (with or without validation), a file drop, an option group or even a set of buttons (with or without pretty pictures).

Once the variable has a value, the following code will ensure that the browse is correctly filtered:

```
...
Case FilterString
Of 'CW2.x'
  BRW1.SetFilter('(Clip(Upper(FAQ:Product)) |
    = 'CW2.X'))')
Of 'IC'
  BRW1.SetFilter('(Clip(Upper(FAQ:Product)) |
    = 'IC'))')
Of 'C4/ABC'
  BRW1.SetFilter('(Clip(Upper(FAQ:Product)) |
    = 'C4/ABC'))')
!repeat for each value FilterString can take
```

```
End  
End  
ThisWindow.Reset(1)
```

If you do not provide a qualified list of choices, the code:

```
BRW1.SetFilter('Clip(Upper(FAQ:Product)) |  
= Clip(Upper(entryField))')
```

will do.

For reports and processes, substitute `ThisReport.SetFilter` and `ThisProcess.SetFilter`, respectively for `BRWx.SetFilter`.

Summary

Most filtering is done with `SetFilter`; this can happen without your being aware of it, since completing the Record Filter prompt results in a call to `SetFilter`.

It is worth examining the code generated by the Record Filter prompt and, for multiple tabs, in `ResetSort` to get a firm handle on how `SetFilter` works. Also, toy with the demo app, commenting out and then uncommenting code in the embeds.

Next time, I'll extend my use of `SetFilter` and show you an extremely powerful use for this method in very complex filter situations.

[Download the source](#)

Notes

1 Variables named in the Record Filter prompt must be `Binded`. This is because variables named in a `SetFilter` statement must be and the Filter prompt creates a `SetFilter` statement. If you fail to do so, you will get an "801" error which will not tell you which variable is causing the problem.

2 It is not, strictly necessary to use the Record Filter prompt to prime the filter. It is also proper to call `SetFilter` directly in `INIT`, after priority 8500.

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 competitors' right side mirrors - while on the track (but only while accelerating). Steve has been writing on Clarion since 1993.

Reader Comments

[Add a comment](#)

Copyright © 1999-2001 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca

Clarion MAGAZINE

Clarion News

[New Clarion Web Server In Development](#)

3D Computing has developed a Web Server for Clarion and is considering releasing this product as a third party development tool to the Clarion community. Any feedback will be welcome. The HTTP Server is constructed from a number of other objects which are also accessible to the developer. It is easily incorporated into your application by plugging in the provided template.

Components include: Http Protocol Layer; Http Session Manager; Http Headers Layer; Cookies Manager; Request Buffering Layer; TCP/IP Communications Layer; Connection Buffering Layer. The TCP/IP communications object can support multiple protocol layers. This allows for future expansion. Expansion possibilities means the end product can be a Web Server as well as an FTP Server, mail server etc at the same time. Session Management allows the use of session variables properly instantiated for each http session. This is transparent to the Developer. Each HTML page served out by the Web Engine is parsed. If the engine finds any developer defined tokens, these are passed via a callback to the developer to be resolved. Future expansion will allow for an interpreted scripting engine which will allow html pages to contain clarion language script which will be executed as the HTML Pages are served out. The sale price of this product is yet to be determined. The pricing may be a royalty basis where the developer can develop with the product with no up front costs. If the developers product sells, 3D will expect a small royalty on each sale.

Posted Friday, April 27, 2001

[PD Lookup Update](#)

Phil Will has posted a new compile the PD File Drop and Drop Combo libraries that fixes an issue in W2K 32-bit which sometimes caused an application to lock up. This was tracked down to a W2K bug that occurs when the numlock key is depressed. The update forces the numlock key off while the file drop or drop combo (string fields) has focus. This issue does not affect the browse lookup button.

Posted Friday, April 27, 2001

[OpenClarion Site Rebuilt](#)

Ron Schofield's Open Clarion web site is back online. New features include a tutorial on installing and configuring Caldera eServer 2.3.1, including a downloadable disk of shell scripts and other files. The Wininet project information page has also been updated.

Posted Thursday, April 26, 2001

[CPCS PDF/Emailer Addon enhancements](#)

The CPCS PDF/Emailer Addon has been enhanced, and now has direct support for the following PDF creation products: Adobe Acrobat 4.0+; Amyuni Consultants PDF Driver Developer's Edition; ZEON DocuCom PDF Creator. Also supported are many other generic PDF printer drivers available via the internet, such as the 5D PDF Creator from Jaws Systems, Ltd. Demo available.

Posted Thursday, April 26, 2001

[PD Translator Plus Supports Ideographic Characters](#)

Phil Will has added some methods to the Translator Plus C55 libraries to support the entry of ideographic characters, such as used in Chinese and Japanese. Just add a global extension and ENTRY controls are replaced with controls that do provide this support.

Posted Thursday, April 26, 2001

[Linder Software's WebUpdate\WebSetup Beta Coming Soon](#)

The new WebUpdate\WebSetup (Web) system interacts with many Linder SetupBuilder functions. Updating and testing these functions have led to the delayed launch date. SetupBuilder's

Web technology allows you to very easily deploy your applications over the Internet. Unlike other web deployment technologies, this product has absolutely no reliance on browser or ActiveX controls. The beta version will be released soon, and registered Linder SetupBuilder customers will be entitled to get the WebUpdate\WebSetup package for a special crossgrade price.

Posted Thursday, April 26, 2001

[New ExceleTel TeleTools Samples Available](#)

Craig Ransom has released an updated version of the TTSamples.zip file, including: a special app to generate a runtime licensing include file has been added; a new template to add the licensing file at the global level and also to create and serial number TeleTools controls at runtime has been added; two sample programs rewritten to use these templates; an updated readme.txt.

Posted Thursday, April 26, 2001

[DC Templates v1.7](#)

Bob Gaucher reports that DC Templates (for data conversion) version 1.7 has just been released.

Posted Thursday, April 26, 2001

[ARCO Word Reporter XP Compatible](#)

Hanspeter Stutz reports that the ARCO Word Reporter has been tested with MS Word XP and is fully compatible.

Posted Thursday, April 26, 2001

[Greenbar Template Gets New Features](#)

Stephen Bottomley has added runtime color setting to his free JaDu Technologies Greenbar templates for ABC and Legacy apps. Demo apps for C55 (with TXA/TXD) are also available. Now compatible with pre 5.0 ABC browses.

Posted Thursday, April 26, 2001

[Free Greenbar Template](#)

Stephen Bottomley has released a free greenbar template for

ABC and legacy browses. Features include: runtime toggle of greenbar; exclude feelds; alternate greenbar colors on different fields; conditional coloring using a mixture of default and conditional colors.

Posted Monday, April 23, 2001

[Solace ColourAll Template Now Free](#)

Simon Burrows has added the ColourAll template to his free template set. This template makes it easy to color all columns in a browse.

Posted Monday, April 23, 2001

[Free Editor Key Configuration Tool](#)

Chris Jong of solid.software has released a free tool that amkes it easy to configure the Clarion editor keyboard shortcuts in Clarion 5 and 5.5. You just press the desired key combinations for the selected editor function.

Posted Monday, April 23, 2001

[Handy Tools Build O-6A Now Available](#)

Gus Creces has released Handy Tools Build O-6A. New features include: full support for user-translatable messages, prompts, etc.; equate prefixing to avoid name clashes with other products; classes incorporate WindowComponent interface where relevant; more complete emulation of Windows marking keystrokes; ListboxBrowseExtender template adds language-independent querying; query parsing supports date and time pictures; export to HTML; slider control synchronization events; instant email template; and much more.

Posted Tuesday, April 17, 2001

[Linder SetupBuilder 3.51 \(Build 6084\) Available](#)

Linder Software has released Linder SetupBuilder 3.51 Build 6084. This release was compiled with Clarion 5.5D. The patch download is less than 1.3MB and requires the Build 6077 release.

Posted Thursday, April 12, 2001



Copyright © 1999-2001 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the expresswritten consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca.