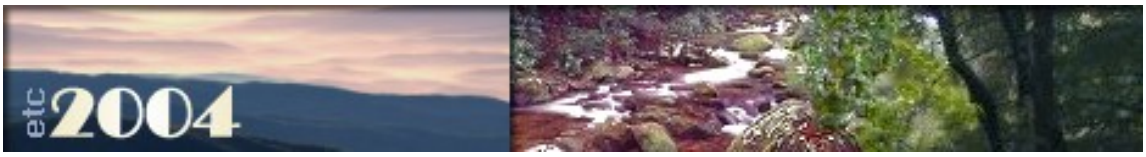
Clarion Magazine



Please note that Clarion Magazine is temporarily publishing three issues per month rather than four so we can devote more time to our Clarion book series. All subscriptions are automatically extended.

Browse Greenbars in Clarion 6

Articles: XML

News: XML

Giving a presentation on Clarion 6 can be a tricky thing, when the topic everyone wants to know about, in this case greenbars, is the one thing you haven't prepared. Under intense pressure and close scrutiny, Carl Barnes and Steve Parker discovered just how this new feature works.

Posted Thursday, February 05, 2004

SQL Identity: Another Approach

MSSQL provides a way of creating autonumbered sysIDs, but what happens when you have special requirements that demand more flexibility? John Griffiths describes an approach using stored procedures that lets you customize your sysID generation.

Posted Friday, February 06, 2004

Book Review: Inside COM

Although Clarion 6 includes greatly expanded COM support, documentation on just how to use COM with Clarion is sparse, to say the least. One book that comes up again and again is this older work by Dale Rogerson. Although out of print, it is still reasonably easy to find, and is a useful reference for anyone doing Clarion COM work.

Posted Friday, February 06, 2004

Weekly PDF for February 1-7, 2004

All articles for February 1-7, 2004 in PDF format.

Posted Tuesday, February 10, 2004

News

Sterling Data Holiday Schedule

xAppWallpaper 1.6

xFunction Library 2.1

Firebird 1.5 Final

Clarion Training In Gauteng Cancelled

Outlook-Related Products

New Clarion Wallpaper

Card Source Demo Updated

RPM Drops Legacy Support

PayWire Credit Card Templates

SoggyNotes 1.2

SURVEY

Do you use drive imaging software as part (or all) of your backup strategy?

Yes, as part of regular backup

7.3%

Yes, after installing OS

7.3%

No, but plan to

21.8%

No, and don't plan to

63.6%

55 responses

Previous Surveys

No Issue This Week

There will be no issue this week (Feb 15-21) as we're busy processing the book orders and getting everything shipped. As usual all subscriptions are automatically extended.

Posted Tuesday, February 17, 2004

Clarion Tips & Techniques Book Pre-Orders Shipped!

All of the Tips & Techniques book pre-orders have now shipped. US orders are shipped by USPS; overseas orders are couriered to a postal center in that country, and then distributed using the postal system.

Posted Wednesday, February 25, 2004

Auto-Complete Files And URLs With SHAutoComplete

Ever wonder how you can add an auto-complete feature like that used in Internet Explorer to your applications? Carl Barnes shows how to use IE's code, via the SHAutoComplete API function.

Posted Thursday, February 26, 2004

Template Styles in Clarion 6

When Clarion 6 first became available to Early Access Program participants, there were many changes to the templates, but there were also some significant changes to the template *language*. Russ Eggen looks at how to use these changes to improve the appearance of your templates, while maintaining backward compatibility.

Posted Friday, February 27, 2004

Creating Derived ABC Classes With a Template

In this article Harley Jones explains some of the behind-the-scenes magic that Clarion performs on ABC windows and controls, and how you can do the same with custom templates.

Posted Friday, February 27, 2004

Firebird 1.5 RC 9

Lindersoft Office Closed Until Feb 16th

BLAT 1.2

Firebird Article

etc2004 Web Site Update

Fenix 1.5 Adds Mobile Support

DevCon 2004 At DisneyWorld In September

SoggyNotes

BST 2.7

SoggyVersion

New API Database & Template

MSDE Usage Conditions Updated

SoggyRunner

Clarion Jobs Page

ImageMan Templates Compatible With C6

Imaging Templates Compatible With C6

1st Shareware Business For Sale

One Year Ago In CM

PDF For February, 2003

Book Review: PostgreSQL Developer's Handbook

Debugging Queues With Excel

Two Years Ago In CM

SV To Announce Upcoming Products

Clarion 5.507 Service Release 7 (Final candidate)

A Closer Look At Required Fields

Three Years Ago In CM

Clarion News - March 2001

The Clarion Challenge: Useless Tab Text

Introduction To SQL: Part 1

Four Years Ago In CM

Clarion News - February 2000

Looking for more? Check out the **site index**, or **search the back issues**. This site now contains more than 700 articles and a total of over a million words of Clarion-related information.

Imaging Templates 1.23 (C6 Compatible)

Replicate 1.34 Beta

ThinkData Releases xmlFUSE 2.1

ThinkData Sale Extended

ABC Free Templates and Tools Updated (C6 only)

SoggyQueue Update

Cards Source Code For Sale

Lodestar/Clarion Add-ons Web Site Redesign

Gitano Software Specials

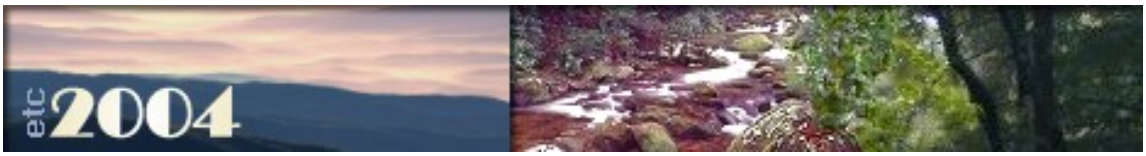
Free Words & Lines Counter

Search the news archive

List Box Marking

Open Source Update: Thread Manager

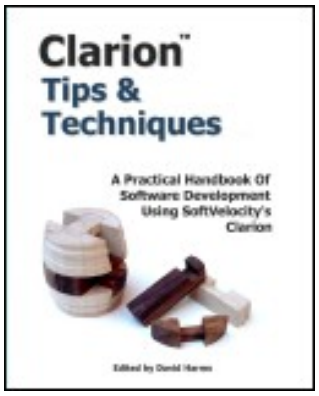
Clarion Magazine



Clarion Books

Clarion Magazine now has books in print! We'll be adding more titles in the near future, so to keep up to date be sure to get on our **mailing list**! If you **already have a free membership or a subscription**, put yourself on the list via the member update form. If you're **not yet registered** with Clarion Magazine, join use the new member form! There's no charge. On either form, select the **Books & Specials** mailing list. This is a double opt-in list, and all emails that go out to you once you've signed up will have an unsubscribe link.

Books in print



Clarion Tips and Techniques

Edited by David Harms

List price: US $89.95
Buy now: **US $76.46**

This 630 page softcover book contains selected articles from Clarion Magazine's first five years in publication, as well as a small number of articles originally published in Clarion Online. These articles were written for Clarion 5.0 and 5.5, but many are just as applicable to Clarion 6.
CoveComm Inc. ISBN: 0-9689553-1-2

See more book details

Buy the book now!

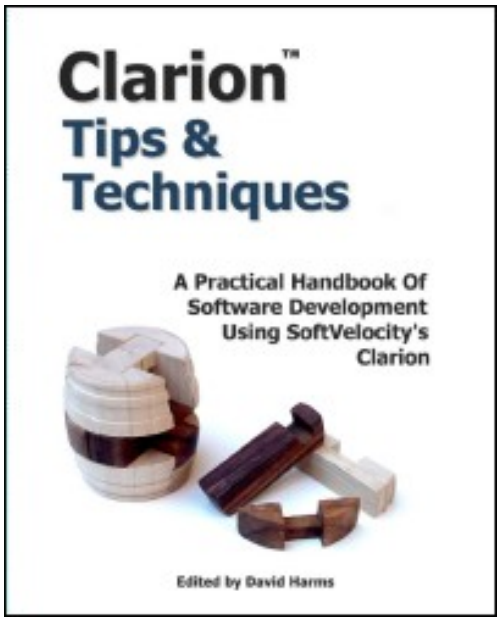
Why is Clarion Magazine printing books?

Clarion Magazine, the online publication, is very popular with Clarion developers, and we're proud of the massive library of information we've published in HTML and PDF form. So why are we now printing books? Because as handy as HTML and PDF documents are, for many of us they still aren't as convenient and, well, enjoyable to read as a printed, bound book.

We are continuing to convert Clarion Magazine's HTML pages to book form. There is a stunning amount of material to choose from; in a 7.5" x 9.25" book format, the Clarion Magazine web site contains some 6500 pages of Clarion articles. That's about two feet of shelf space! If there is sufficient interest we will consider publishing all of this material, but it would be impossible to bring it all to press in a short period of time. As a result, we're focusing our efforts on areas likely to be of most immediate interest to Clarion developers. Stay tuned for more book news! And if you have any questions just send us an email.

Clarion Magazine

Clarion Tips & Techniques Book



Clarion Tips and Techniques is 630 page softcover book, containing selected articles from Clarion Magazine's first five years in publication, as well as a small number of articles originally published in Clarion Online. These articles were written for Clarion 5.0 and 5.5, but many are just as applicable to Clarion 6.

CoveComm Inc. ISBN: 0-9689553-1-2

Buy the tips book now!

Source code

The source code for the Clarion Tips & Techniques book is contained in the following zip file:

ClarionTipsTechniquesSource.zip (3.86 MB)

Errata

Corrections to the book will be posted here as necessary.

Frequently Asked Questions

Q. How is the book printed?

A. Clarion Magazine's books are trade paperbacks with color covers, and are produced using print-on-demand (POD) technology. The interior of the book is printed on high quality, acid-free, book grade opaque paper. The printing service we use has produced more than 10 million books this way, for small and large publishers.

Q. Can I view any part of the book online?

A. Yes. You can view the Table of Contents (PDF, 70 KB), Index (PDF, 118K), and Author

Index (PDF, 101KB) online. If you have difficulty downloading any of these PDFs please try right clicking on the link and choosing Save Target As (or equivalent).

Q. Do I need a subscription to buy this book?

A. No, a subscription is not required, but you have not previously registered with Clarion Magazine you will be asked to do so during the purchase process. We keep your information confidential, in accordance with our privacy policy.

Q. Do you keep my credit card information on file?

A. We do not keep any customer credit card information. In fact, we don't normally even see it, since all online credit card transactions are processed on our behalf by InternetSecure, at their site.

Refund policy

If you are not satisfied with the book you have 30 days from shipment date to return the book, in new condition, to CoveComm Inc, publisher of Clarion Magazine, for a full refund or credit of the book purchase price.

Clarion Magazine

Browse Greenbars in Clarion 6

by Steven Parker and Carl Barnes

Published 2004-02-05

It’s funny what people find interesting or important.

We (Carl and Steve, Steve and Carl) were presenting C6 to the Chicago user group, and the built -in greenbar effect on browses, by popular acclaim, captured people’s attention for almost half an hour. ADO, sorting browses by column headers, the revised threading model and the magnificent new rebasing template drew choruses of "that’s nice; how’s that greenbar thingee work?" Actually it was more of a heckle, as in "Did they ever get a working greenbar?"

So, in front of God and everyone, we had to make it work. Neither of us had ever seen it before, but hey, Clarion’s easy. We probably just have to check a box.

Okay, PEOPLE.APP in the examples has a nice browse to work with (we’d just finished using it to show the column sort feature). Let’s see, where’s that check box for enabling greenbar? Must be in the browser extension template properties? Nope. Then there must be an extension template we have to add? Nope.

Back to the drawing board. Someone suggested maybe it was time to read the manual. Bob Foreman must have documented it. The help and PDFs with C6 are a big improvement over the already fine documentation. The help index didn’t have "green" anything. But entering "greenbar" on the help search tab returned six results.

It turns out that the greenbar effect depends on the "Listbox Styles" used to conditionally set fonts and colors for individual listbox cells. Which means it is worth taking a look at how to implement styles and the greenbar effect. So, without further ado …

The hardest part of implementing the greenbar effect turned out to be implementing styles for the listbox. And, the hardest part of implementing styles is finding where styles are set/selected. It took us 10 minutes to find it. It’s always the large, clearly labeled buttons that are the hardest to find.

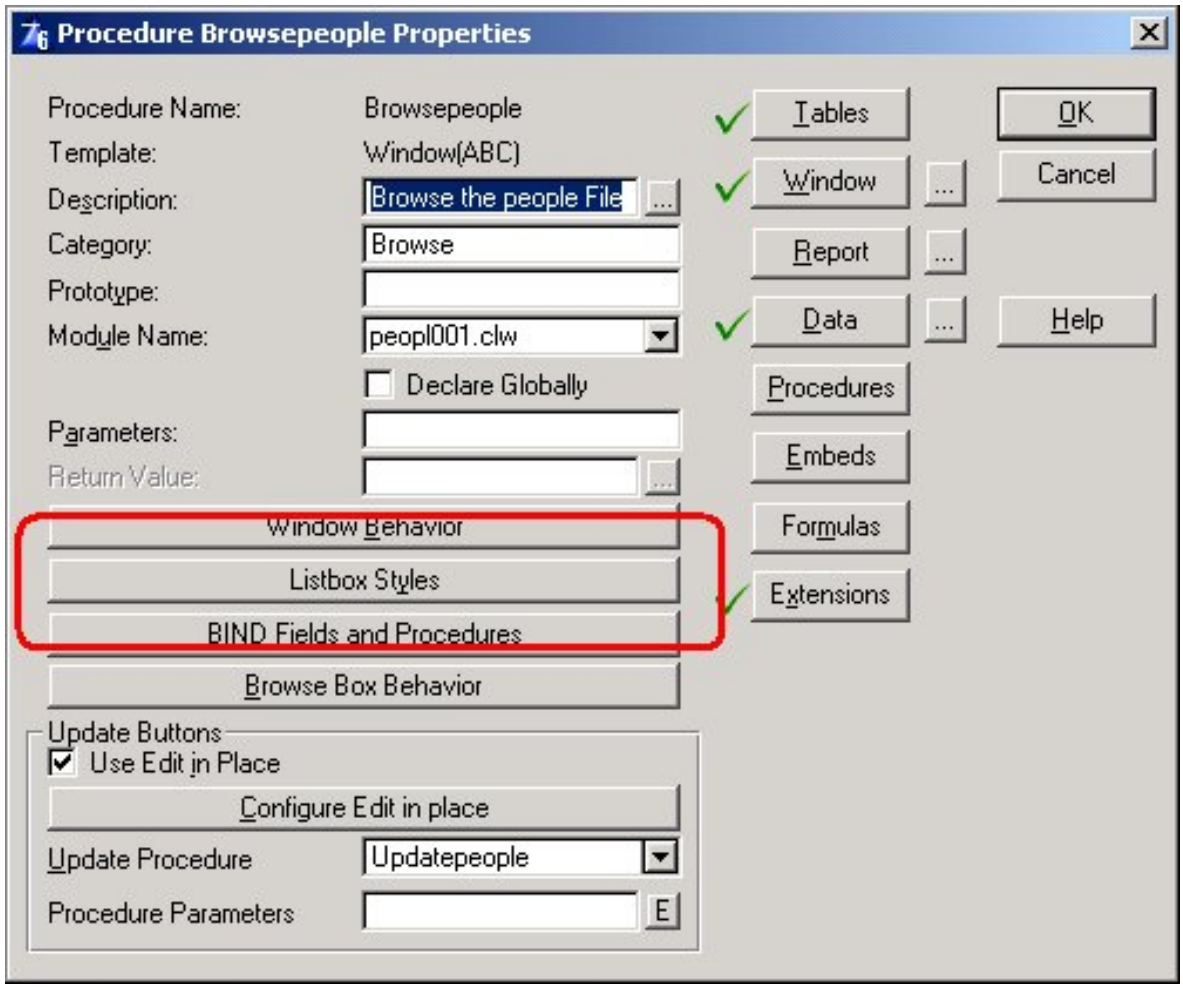


Figure 1. Entry point for setting listbox styles

Having conquered the monumental task of finding where styles are set, two styles are necessary. One is for the regular lines and one is for the green lines (you're not actually constrained to use green, you could use fuchsia if you really want to).

The screenshot below shows that the regular lines using default colors (`Color:None`). This makes the assumption that the user will have default colors that work well with the "green" line colors we pick. To assure a consistent look it would be best to explicitly set all colors. We didn't do it because we think users who pick odd colors deserve an ugly screen.

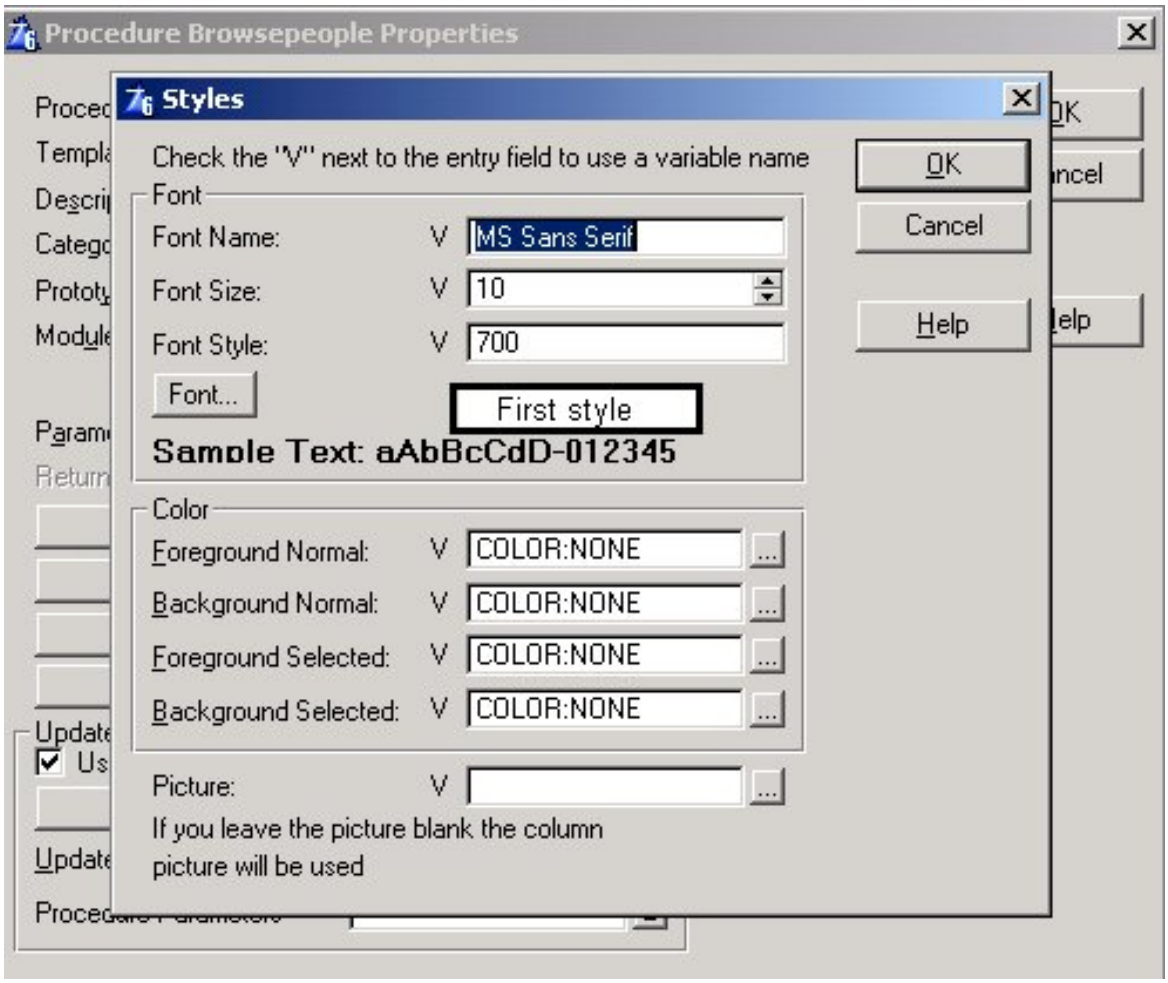


Figure 2. Setting the style for the regular lines

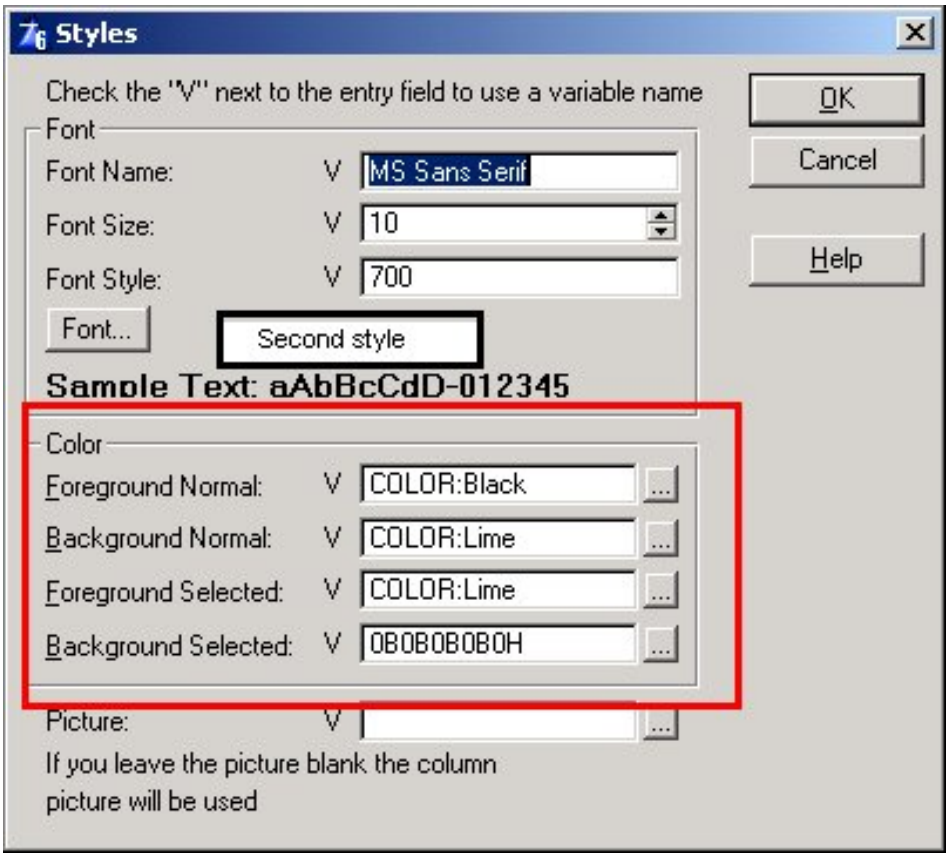


Figure 3. Setting the style for the green lines

After the two styles are defined, the styles list contains two styles. Note that they are numbered:

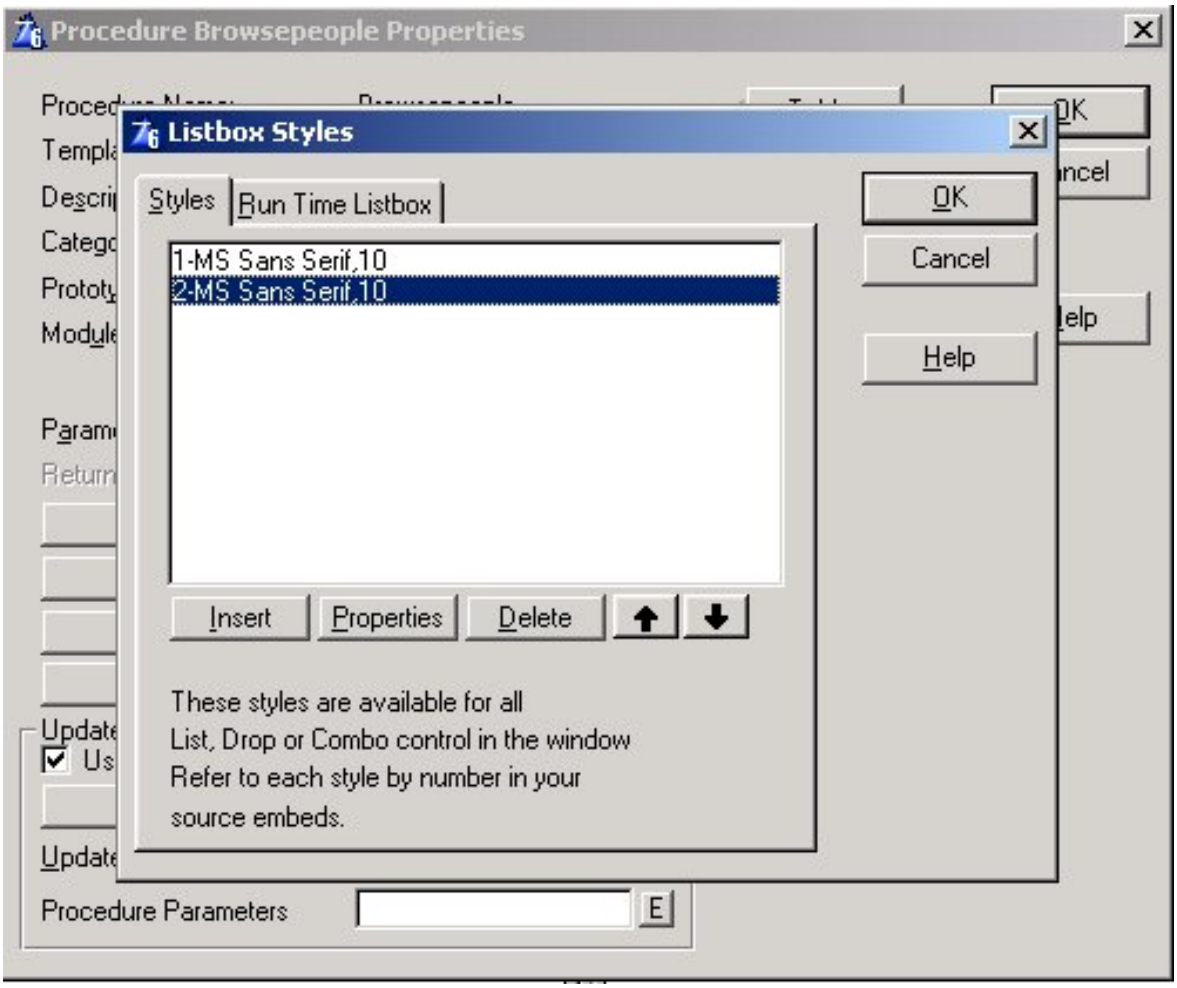


Figure 4. Styles list

To notify the list that you are using styles, you must check the Styles checkbox. Right click on the listbox and select List Box Format. In the lower right corner is a Styles checkbox.

The Styles box must be checked for *each and every* column in the list.

Now right click the listbox again and select Actions, then select the Styles tab. The screen will not look like the below screen shot until you check Use same Style for All Columns and check Create Greenbar effect..

Once Create Greenbar Effect is checked, the two styles group boxes are enabled. Simply select the two local styles configured earlier.

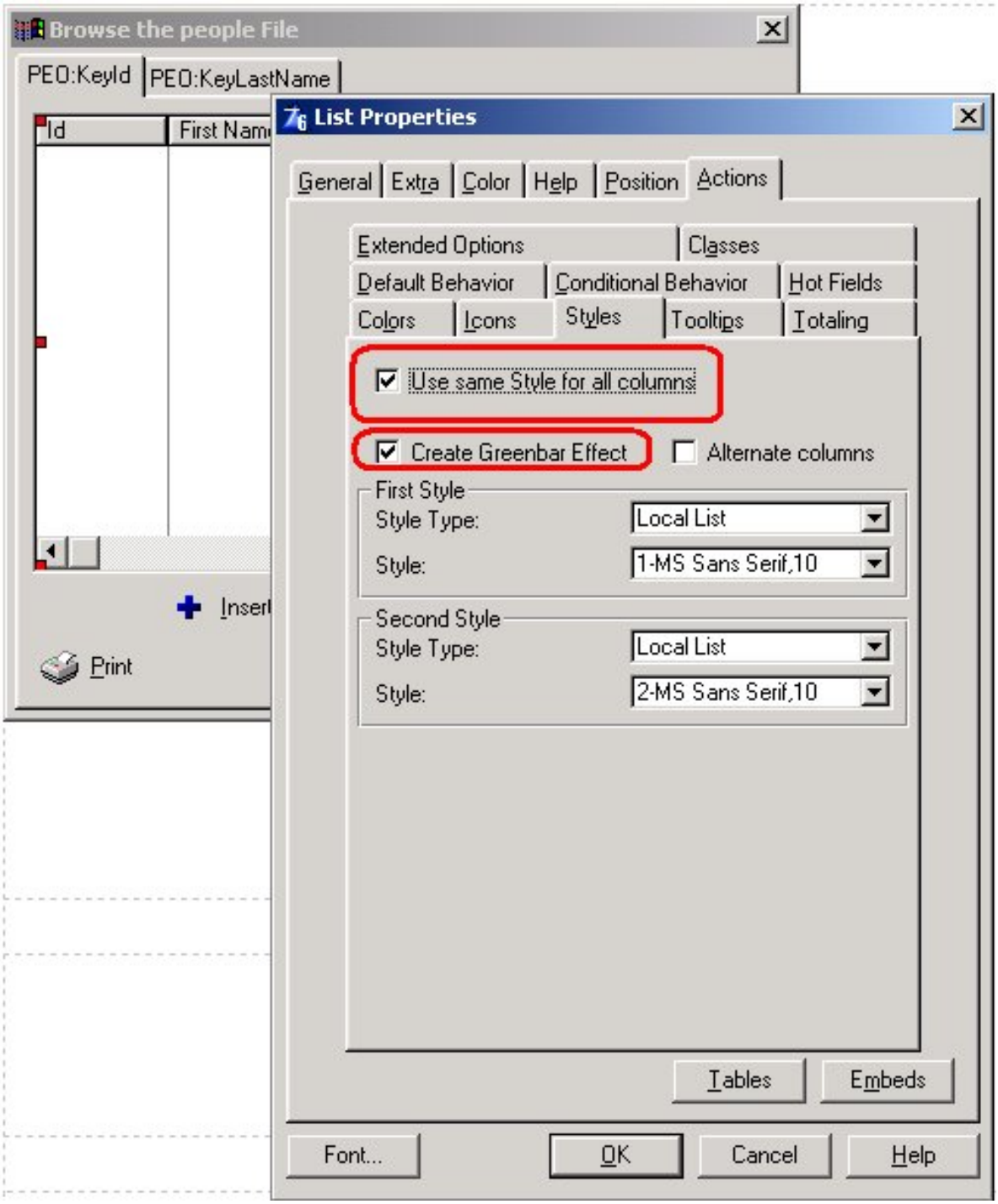


Figure 5. Actions implement the green bar effect

If you run the app (we have provided a locally linked EXE if you don't yet have C6), you will see a browse like this:

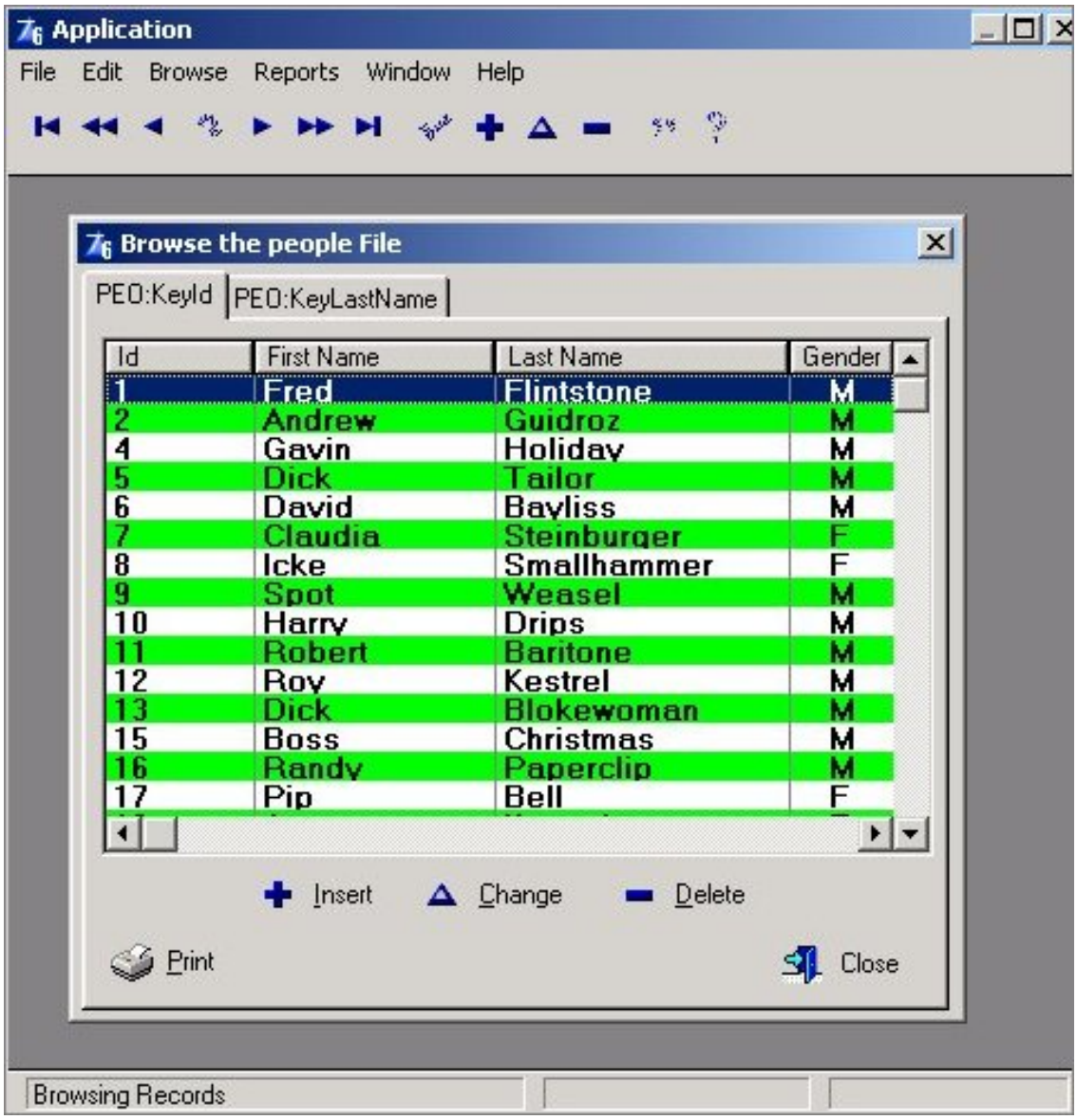


Figure 6. Greenbar effect in effect

You may have noticed an Alternate Columns checkbox to the right of the Create Greenbar effect checkbox on the Styles tab.

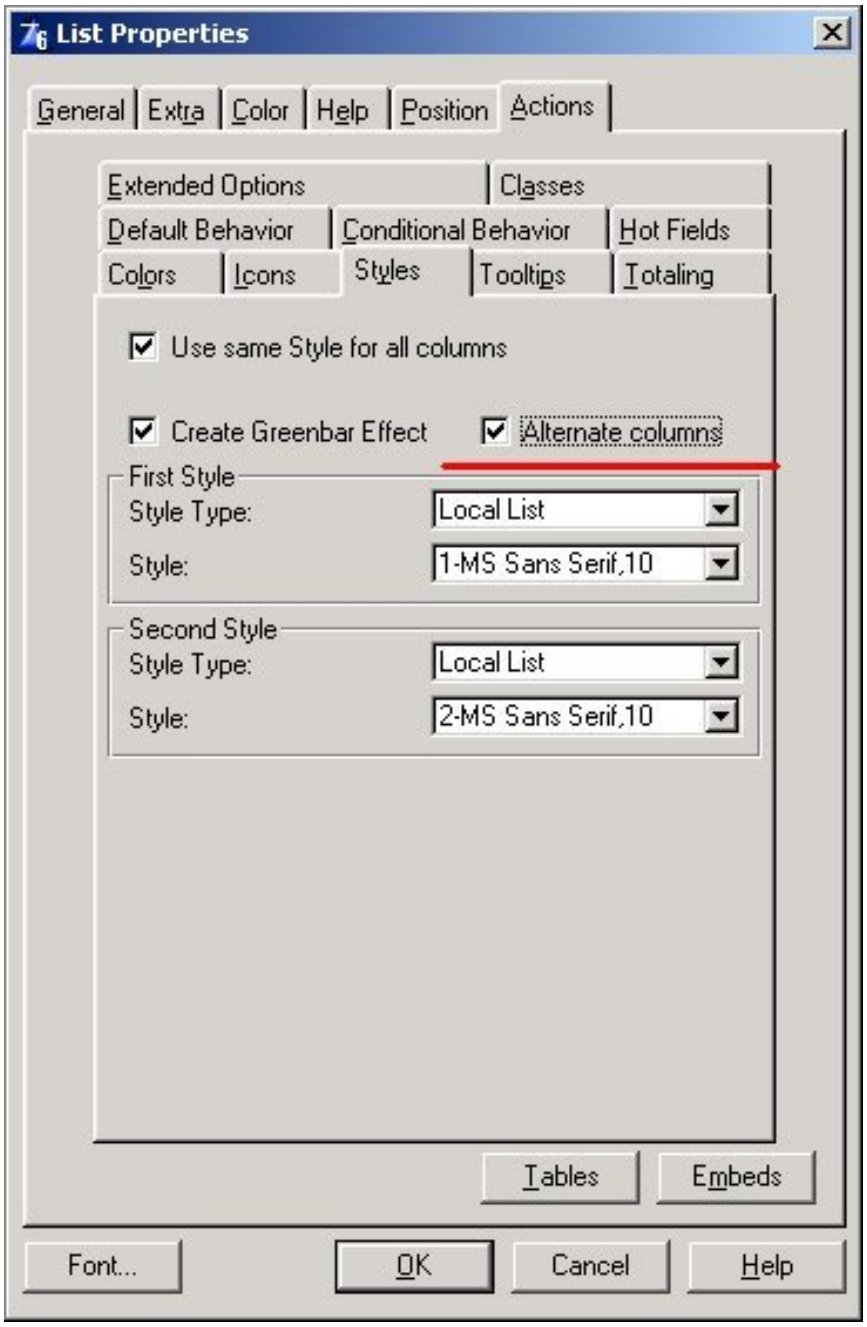


Figure 7. Alternate Columns

(One of the authors, who shall go nameless [but Steve says it wasn't him], discovered the Alternate Columns checkbox and became somewhat enthusiastic over it.)

Before checking this, take a dose of Dramamine:

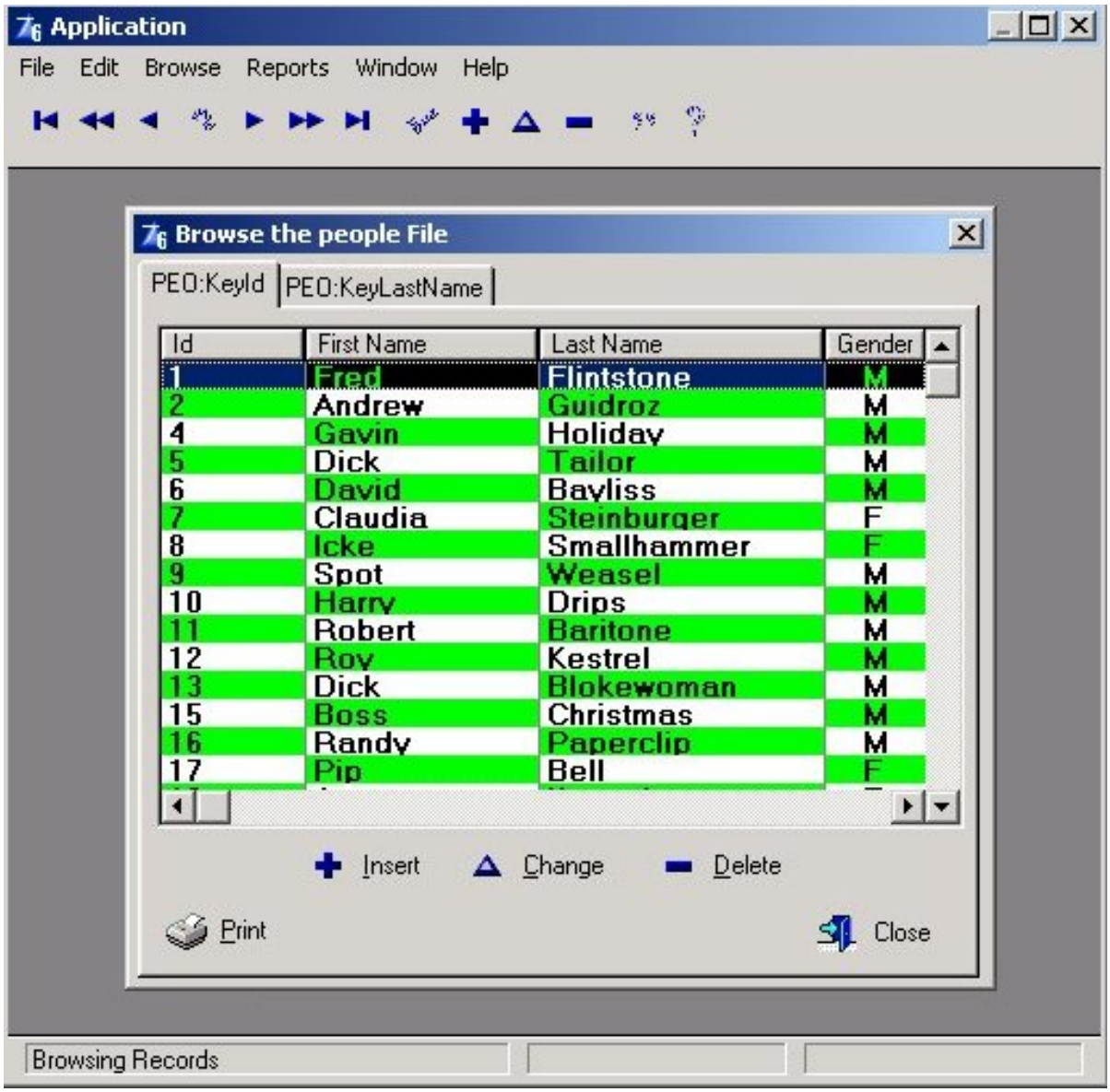


Figure 8. Alternate Columns in action

And, because you don’t have to use a style on every column, you can really customize the look of your browse.

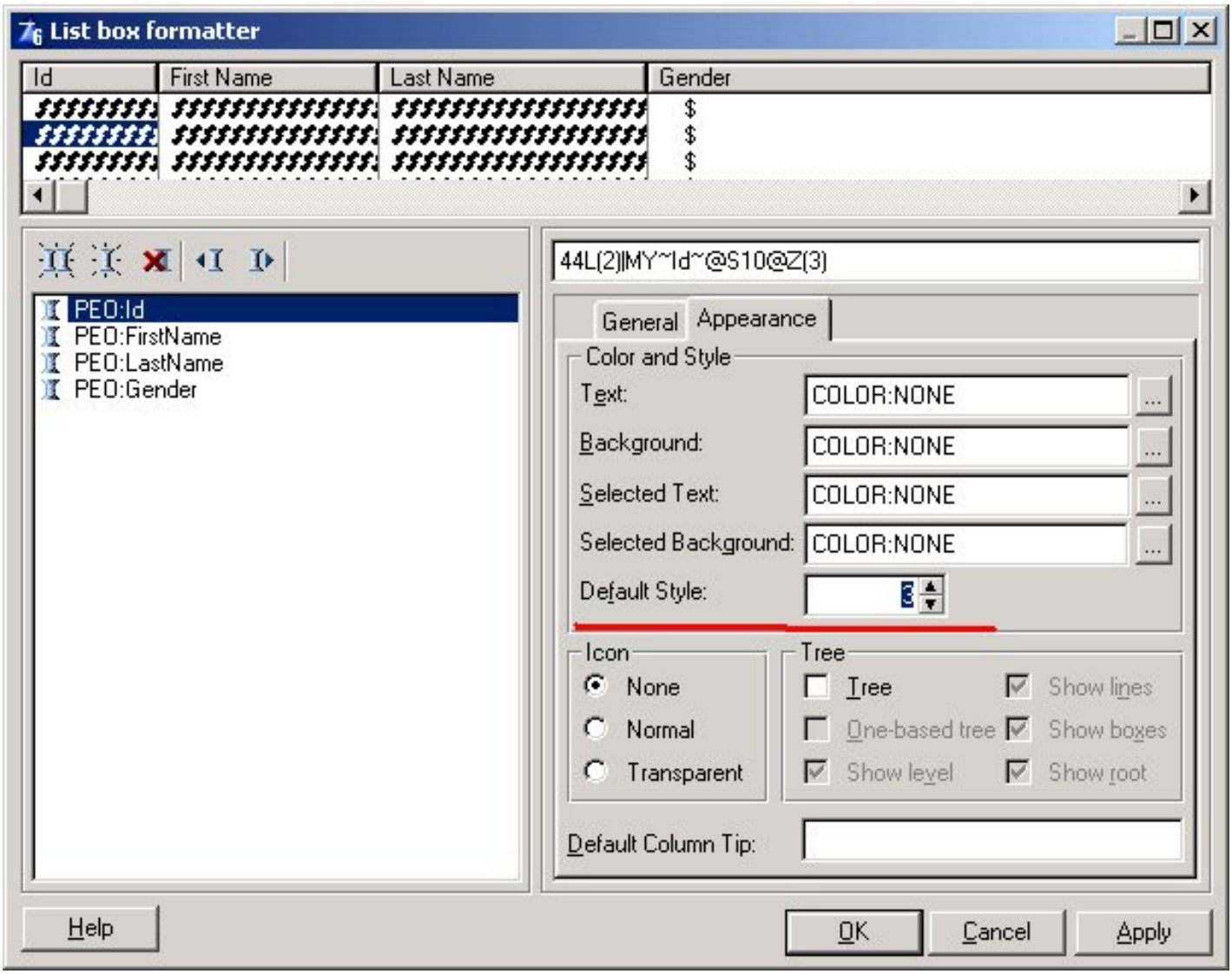


Figure 9. Setting a style for a single column

The source code

A style in a Clarion listbox is implemented somewhat like an icon. A `LONG` is placed in the list queue immediately following the column that will use the style. This `LONG` will contain the style number the column will use for display. The browse queue generated by the templates from our example is shown below and has a style `LONG` after each column:

```
Queue:Browse:1    QUEUE
PEO:Id              LIKE(PEO:Id)
PEO:Id_Style        LONG                !Field style
PEO:FirstName       LIKE(PEO:FirstName)
PEO:FirstName_Style LONG                !Field style
PEO:LastName        LIKE(PEO:LastName)
PEO:LastName_Style  LONG                !Field style
PEO:Gender          LIKE(PEO:Gender)
PEO:Gender_Style    LONG                !Field style
                  END
```

And also like listbox icons, the styles for the listbox are defined using properties of the list control. The style definition code generated by the templates from our example is shown below:

```
DefineListboxStyle ROUTINE
  ?Browse:1{PROPSTYLE:FontName, 1}      = 'MS Sans Serif'
  ?Browse:1{PROPSTYLE:FontSize, 1}      = 10
  ?Browse:1{PROPSTYLE:FontStyle, 1}     = 700
  ?Browse:1{PROPSTYLE:TextColor, 1}     = -1
  ?Browse:1{PROPSTYLE:BackColor, 1}     = -1
  ?Browse:1{PROPSTYLE:TextSelected, 1}  = -1
  ?Browse:1{PROPSTYLE:BackSelected, 1}  = -1
  !----------------------------------
  ?Browse:1{PROPSTYLE:FontName, 2}      = 'MS Sans Serif'
  ?Browse:1{PROPSTYLE:FontSize, 2}      = 10
  ?Browse:1{PROPSTYLE:FontStyle, 2}     = 700
  ?Browse:1{PROPSTYLE:TextColor, 2}     = 0
  ?Browse:1{PROPSTYLE:BackColor, 2}     = 65280
  ?Browse:1{PROPSTYLE:TextSelected, 2}  = 65280
  ?Browse:1{PROPSTYLE:BackSelected, 2}  = -1330597712
```

To make a column use a style is simply a matter of setting the `LONG` style code in the queue to the style number you desire. Below is the generated code that sets the entire list queue in one loop to alternating styles:

```
 LOOP GreenBarIndex=1 TO RECORDS(SELF.Q)
   GET(SELF.Q,GreenBarIndex)
   SELF.Q.PEO:Id_Style = CHOOSE(GreenBarIndex % 2,1,2)
   SELF.Q.PEO:FirstName_Style = CHOOSE(GreenBarIndex % 2,1,2)
   SELF.Q.PEO:LastName_Style = CHOOSE(GreenBarIndex % 2,1,2)
   SELF.Q.PEO:Gender_Style = CHOOSE(GreenBarIndex % 2,1,2)
   PUT(SELF.Q)
 END
```

All of this generated code has before and after embed points that allow you to customize or override the template code.

A global approach

If you like the look of the greenbar effect you will probably want to implement it in multiple browses. But using the IDE Listbox Styles button to define your styles in every single browse procedure would be a bad idea. Changing your mind (or a user request color change) would require editing the listbox style settings in every browse.

A method to globally define and set styles would be a better approach. The greenbar settings on the Style tab have a drop list that allows changing the style type from Local List to Style Number as shown below:

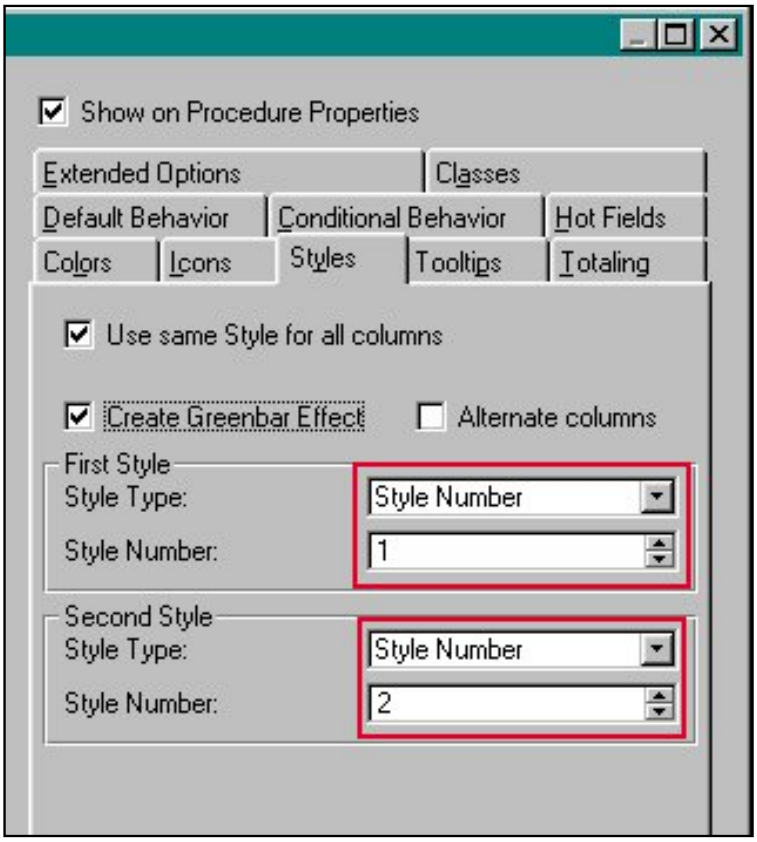


Figure 10. Setting Greenbar Style Number Directly

Then it's up to you to write the `PROPSTYLE:` code to define all of the elements of the style in the `DefineListboxStyle` routine. For globally defined styles you could have all browses call a procedure to set all of your greenbar styles. Then you could even implement it so individual users could decide on their own colors or turn off greenbars by setting all of the colors to `Color:None` (-1).

Source from the included app `BrowseGlobalStyle` procedure is shown below. Note that the font related style properties are not defined so the list font will be used.

```
DefineListboxStyle ROUTINE
  GreenBarColors(?Browse:1)

GreenBarColors       PROCEDURE  (LONG LBFEQ)
  LBFEQ{PROPSTYLE:TextColor, 1}     = COLOR:Black
  LBFEQ{PROPSTYLE:BackColor, 1}     = COLOR:White
  LBFEQ{PROPSTYLE:TextSelected, 1}  = COLOR:White
  LBFEQ{PROPSTYLE:BackSelected, 1}  = COLOR:Navy
  LBFEQ{PROPSTYLE:TextColor, 2}     = COLOR:Black
  LBFEQ{PROPSTYLE:BackColor, 2}     = COLOR:Lime
  LBFEQ{PROPSTYLE:TextSelected, 2}  = COLOR:White
  LBFEQ{PROPSTYLE:BackSelected, 2}  = COLOR:Navy
```

Summary

That's how you implement styles and the greenbar effect. And, because you can set style codes for rows, columns and individual cells, you can really customize the look of your browse. For example, you can

make pay check amounts that are negative show in red and bold. Styles have been in the Clarion language since C5 but not well supported by the IDE. C6 adds full IDE support and a number of template changes that make using styles easy.

Download the source

Steve Parker started his professional life as a Philosopher but now tries to imitate a Clarion developer. He has been attempting to subdue Clarion since 2007 (DOS, that is). He reports that, so far, Clarion is winning. Steve has been writing about Clarion since 1993.

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

Reader Comments

Add a comment

Is it easy for the user to turn on/off the greenbar...
re: Is it easy for the user to turn on/off the...
Am I missing something or is the generated greenbar code...
I agree it is verbose and does not work like the rest of...

Clarion Magazine

Reborn Free CLARION online

SQL Identity: Another Approach

by John L Griffiths

Published 2004-02-06

Having researched Identity field management from the Client-Side (in Clarion) and the Server-Side (in MS-SQL 2000) I determined that neither worked adequately in a project I was working on. The specifications called for inserting a series of child records at the same time as the parent record was inserted, i.e. multiple task records for each job record (JOB <-->> TASKS). There was also another need to set each sysID into a certain range depending on something called the `branchID`. In that project each sysID was set as a `DECIMAL(12,0)` and held the three digit branch number in the billions area. Thus for a branch numbered 333, IDs would range 333,000,000,001 through 333,999,999,999. The long term plan was to move data from all branches into one database and continue to be able to do inserts which retained the numbering sequences.

From that project came the method I describe here for managing the sysIDs of each table within the SQL database. The method described here is simplified by ignoring the `BranchID` feature and uses integer fields for the sysIDs. The sysID fields in the database tables do not have the `IDENTITY` property.

Creating the sysIDs

Basically, the SQL database manages the supply of the sysIDs. The program simply says "Give me the next number for this table".

The system uses a table, called `nextNumTbl`, with two fields, `nextnum` and `tblname`.

When the Clarion program needs to do an insert, it calls a stored procedure in the database that supplies the next ID number for the table specified in the call. Each call to the stored procedure will return a unique unused ID for the table specified. A return of zero will signify an error.

The stored procedure works by selecting the appropriate nextnum field from the `NextNumTbl`, increments the ID and tries to update the `NextNumTbl`. If it succeeds, it returns the ID and the `NextNumTbl` will now hold the next available number.

This process makes it easy to manage inserts of child records without having to rely on the SQL `IDENTITY` processes; `@@IDENTITY`, `IDENT_CURRENT()` or `SCOPE_IDENTITY()`. Your program then has the parent ID and any child IDs needed for data integrity.

In my SQL tables, each table that needs managing this way has a field named `sysID` of type `int`.

The NextNumTbl has a row for each table. Here is the CREATE statement for NextNumTbl:

```
create table dbo.nextnumtbl (
  nextnum  int  null,
  tbl      varchar(20)  not null) --adjust:widen  to suit
```

Data in the table may look like this:

2091	ACNOTE
7123	JOB
59003	TASK
1873	CLIENT

The stored procedure call

I need to call the stored procedure anytime I need to insert…

Here is a call from within Clarion to fetch and prime the next sysID for the acnote table

```
tblName  = 'ACNOTE'
tblName = UPPER(tblName)
SQLcallString = 'CALL jgNextID ( ' & TblName & ')'
    TempF1{prop:sql} = SQLcallString
NEXT(TempF1)
GotSysid = TempF1.F1
IF GotSysid > 0
  ACN:Sysid  = GotSysid
ELSE
  ACN:Sysid  = 0
  ! Handle error here
END
```

Note: TempF1 is a dummy table with one column of type CSTRING, and TblName, SQLcallString and GotSysid are local variables.

The jgNextID Stored Procedure

The jgNextID stored procedure is called with the name of the table as a parameter. It looks up the next number and returns the number via the dummy table buffer.

Here is the CREATE statement for this stored proc, with notes below describing significant lines:

```
1 CREATE PROCEDURE jgNextid ( @tblName varchar(20) )
2 AS
3 DECLARE @nextid  int  ,
```

```
4         @maxtimes int
5 set nocount on
6 set @maxtimes =10
7 while (@maxtimes > 0)
8 begin
9        SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
10     begin transaction
11       -- get the next number to use
12     SELECT @nextid = nn.nextnum from nextnumtbl nn
13            where nn.tbl = @tblName
14      -- Bump figure in DB
15      update Nextnumtbl
16          set nextnum = @nextid + 1
17          where    tbl = @tblName
18            and nextnum = @nextid
19      if @@ERROR = 0
20        begin
21          commit tran
22          break
23        end
24      else   -- if error
25         begin
26         rollback tran
27         set @maxtimes = @maxtimes - 1
28         continue     -- loop again
29         end
30 end  -- while Begin
31 if @maxtimes < 1
32   set @nextid = 0
33 select ( @nextid )
34 set nocount off     -- reset it
```

Line 6: Set the number of tries the proc will use to get a valid number. With a site having twenty users inserting data and a value here of 5, I have not seen any collisions

Line 9: Set isolation level to protect the transaction

Lines 12,13: The select statement to fetch the next available number

Lines 15-18: Attempt to update the table where the nextnum matches the value just fetched. This ensures a good number held by no one else.

Lines 19-23: If the update was successful, break from the while loop.

Lines 24-29: The update failed, so continue loop.

Lines 31-32: If the loop counter hit zero, set `@nextID` to 0

Line 33: Select the value in `@nextID` for collection by the Clarion program.

Management stored procedure

What if you want to reset the starting number for the next sysID? To manage the `NextNumTbl` another stored proc was developed. This one is called for each table and sets/resets the `nextnum` field as required. Again, see the notes below for what is happening.

```
1 CREATE PROC jgSetNextNumFor( @tblName  varchar(20))
2 AS
3 DECLARE         @nextInt integer  ,
4                 @gotMax  integer  ,
5                 @selStr   varchar(201)  ,
6                 @currentNextInt   integer
7
8 SET @selStr='SELECT n1 = MAX(sysid)INTO ##JJ8 FROM '
9              + @tblName
10 BEGIN
11   EXEC (@selStr )
12   SELECT @gotMax = n1 FROM ##JJ8
13   DROP TABLE ##JJ8
14 END
15 IF @gotMax = 0 OR @gotMax is NULL
16     SET @nextInt = 1
17 ELSE
18     SET @nextInt = @gotMax + 1
19
20 SELECT @currentNextInt = nextnum FROM NextNumTbl
21        WHERE tbl = @tblName
22 IF @currentNextInt is NULL OR @currentNextInt=0
23    BEGIN
24      INSERT INTO NextNumTbl VALUES (
25        @nextInt ,
26        @tblName  )
27      RETURN
28    END
29 IF @currentNextInt < @nextInt
30    UPDATE NextNumTbl
31      SET nextnum = @nextInt WHERE tbl = @tblName
```

Lines 8-9: Build a string needed for the select statement at line 11. As that `EXEC` call will be in a separate scope, the local variables are not visible so their values need to be put into the select string. Table `##JJ8` is created as a global temporary table.

Line 11: This will execute the select string and places the result into a field named N1 in the global temporary table named ##JJ8 (These temporary table names are not significant – use whatever you like)

Line 12: Collect the table's maximum value here and place it into the local variable.

Line 13: Drop the temporary table.

Lines 15..16: If the result is a 0 or null value, it shows that there were no records for that table, so set the next

number to 1

Line 18: Bump the max value by 1

Lines 20-21: See what value exists in the `NextNumTbl` for the table specified.

Lines 22-28: If there is no valid number there, go ahead and insert one.

Lines 29-31: If the number available in the `NextNumTbl` is less that the `MAX (+1)` value in the table, then update `NextNumTbl` with the next valid number.

Extension Template

Here is an extension template for adding to standard forms. It will do the necessary field priming for the table inserts. This is my first foray into template programming, but it works!

```
#TEMPLATE (JGsql4, 'SQL Table Identity Aid by JLG')
#Extension(JGAutoIncSysid, 'For insert priming from StoredProc jgNextNum')
#boxed('SQL JG SP_ AutoInc')
#DISPLAY('Enter the table Clarion prefix:-')
#PROMPT('  Table Prefix Code:',@S3),%TablePre
#DISPLAY('Next name the SQL table:-')
#Prompt('  Table for Insert ' ,@s20),%JGFile
#DISPLAY('')
#endboxed
#at(%DataSectionbEFOREwINDOW)
tblName      CSTRING(21)
GotSysid     LONG,auto
SQLCallString   cstring(401),auto
#endat
#AT(%PrimeFields,'Prime record fields on Insert'),WHERE(%InsertAllowed)

#DECLARE(%JLGTblSysid)
  tblName = '%JGFile'
  tblName = upper(tblName)
  SQLcallString = 'call jgNextID ( ' & TblName & ')'
  Open(tempF1)
  TempF1{prop:sql} = SQLcallString
  next(TempF1)
  GotSysid = TempF1.F1
  #set(%JLGTblSysid,%TablePre & ':' & 'Sysid')
  if GotSysid > 1
    %JLGTblSysid  = GotSysid
  else
    %JLGTblSysid  =  0 !and insert should fail
  end
  Close(TempF1)
#EndAT
```

Summary

This method of handling identity values for child inserts from within a parent record has worked well in a large multi-DLL app which was moved several years ago from Clarion DAT files to MS-SQL2000. While it takes a little extra setup work, it simplifies sysID management for child inserts. Thanks go to George Hale for insight on how this could be achieved.

John Griffiths, an Australian, has been developing with Clarion since the DOS days. John has two summers each year, spending six months in Australia for the Southern summer and six months in Texas for the Northern summer. He works as a contractor for a Texas company and has developed several business/financial programs which he sells worldwide. John has a B.Bus degree with a major in Informations Systems.

Reader Comments

Add a comment

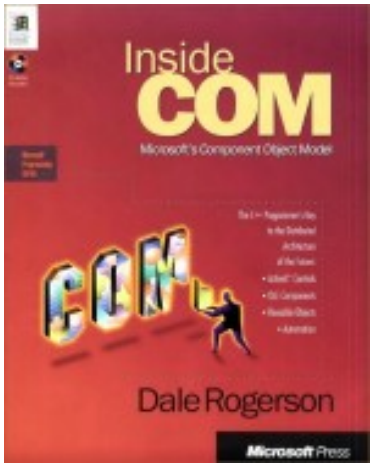
Hi John This is REALLY interesting; I explored a...
Nick said "With the tps driver, the new record is...

Clarion Magazine

Book Review: Inside COM

by David Harms

Published 2004-02-06



Inside COM

by Dale Rogerson
Published by Microsoft Press, 1997
ISBN 1-57231-349-8
376 pages, US $34.99 (used prices vary)

I should mention right at the start that this book is out of print. And that suggests that it is outdated and/or unavailable. Neither of these is true. Although the book was published in 1997, it still accurately describes the fundamentals of COM. And you can find used copies for sale at Amazon and other online booksellers, so it's not that difficult to get your hands on it (if nothing else, try your local library). There are other (and more current) books on COM; still, this book interested me because it's been recommended by developers like Clarion Magazine's guru of things COM, Jim Kane, and SoftVelocity's Pierre Tremblay, who has stated that reading this book will make the SoftVelocity COM code a lot more understandable. The code that this book will help explain can be found in the C6 source files svcomdef.inc and svcom.inc/clw, which are in the libsrc directory.

COM as a technology has been around for a decade, and available to Clarion developers, in some form, for several years, as Jim Kane's C5/5.5 COM articles attest. With Clarion 6 COM support is considerably improved, and in particular shows up in the new ADO classes. Now, you will hear talk that COM is passe, having been eclipsed by .NET technologies. I think that talk is a bit premature, and just as operating systems tend to hang around for a long time, so do fundamental programming technologies (after all, there are still a lot of Clarion developers writing and maintaining legacy applications!).

So take it as read that COM is alive and well. But what is COM, really? As Rogerson points out in his introductory chapter, COM is not a language, but a way of programming. You can (in theory, anyway) write COM components in any language, as long as those components adhere to the rules of COM. And basic to those rules is the concept of an interface.

Rogerson's core definition of a COM interface is "a specific memory structure containing an array of function pointers." Roughly, that means that in order to use any COM component, you need to have a memory structure, in your language of choice, that corresponds exactly to the COM object's interface. There are a lot of details here that matter, and Rogerson covers them carefully. He does use C++ examples, but if you're comfortable with object-oriented programming, and even vaguely familiar with C++ syntax, or have a C++ basics book handy, you should be able to get by.

After dealing with the basics of COM interfaces, Rogerson goes on to discuss how to query a component for which interfaces it supports, using the `IUnknown` interface, which every COM component must inherit from. `IUnknown` defines three methods which are the key to using COM: `QueryInterface`, `AddRef`, and `Release`. You use the first of these methods to ask the component if it supports a specific interface, and the other two methods keep track of the number of interface instance references you've used, so that the component can clean up memory when you're done. And if you've wondered how all of this is implemented in the COM component itself, there are some simple C++ examples you can follow.

So far, COM sounds pretty easy! But of course there is more, and for the first hundred pages or so the author is very deliberately hiding some details to keep the topic manageable. With chapter six he begins to discuss some of these topics, such as `HRESULT`s, `GUID`s and the registry. `HRESULT`s are result or status codes, which may be standard or customized. `GUID` stands for Globally Unique Identifier, which is something every COM component must have. These are implemented as 16 byte (128 bit) numbers, made up of a hardware identifier (typically the network card's address) and a timestamp, and if you're creating (not just using) a component there are utilities that will generate `GUID`s. This chapter also introduces the `CoCreateInstance` function, which takes a component's `GUID` as a parameter, and returns the name of the file (typically a DLL) containing the component. How does it do that? By using the registry. `ProgID`s, which are shorthand, easier-to-use aliases for `GUID`s, can also be found in the registry. In fact, I found the few pages on COM and the registry quite illuminating.

There are a number of COM library functions that make creating and managing components easy, or at least easier. These include `CoCreateInstance`, which behind the scenes uses a class factory to create the component. For more flexibility, you can also work with the class factory directly (it's not clear to me how you would do this in Clarion, although I notice there is an `_IClassFactory` group declared in libsrc\svcomdef.inc).

There is a full chapter on the `IDispatch` interface, which offers an alternative way of communicating with a COM component. Instead of getting a reference to an interface and calling functions, you pass the name of the function to the `IDispatch.Invoke` method, along with any parameters. This can get pretty complicated, since parameters are passed as

variant types. (For the Clarion equivalent, see the `gVariant` group in svcomdef.inc – in fact, you might want to print that file out for reference as you go through this book.) BSTRs, SAFEARRAYs, and type libraries are all discussed. Again, Jim Kane has written about `IDispatch`, although these articles are not quite up to date regarding Clarion 6.

The penultimate chapter explains how COM functions in a multi-threaded environment, and the book concludes with a discussion of a larger example application.

Although you will want some familiarity with C++ to get the most out of this book, it's a well-written treatment of COM basics, and a useful resource for anyone wanting to get the most out of Clarion 6's COM capabilities, where documentation is currently lacking.

Resources

- Dr. GUI Articles at MSDN

David Harms is an independent software developer and the editor and publisher of Clarion Magazine. He is also co-author with Ross Santos of Developing Clarion for Windows Applications, published by SAMS (1995). His most recent book is JSP, Servlets, and MySQL, published by HungryMinds Inc. (2001).

Reader Comments

Add a comment

Clarion Magazine

Clarion News

Search the news archive

Sterling Data Holiday Schedule

The Sterling Data office will be closed from Feb 26 until March 11. Email support will be intermittent, but purchases can be made as usual at ClarionShop.
Posted Tuesday, February 24, 2004

xAppWallpaper 1.6

SealSoft has released xAppWallpaper version 1.6. This release has a new menu item and button control template.
Posted Tuesday, February 24, 2004

xFunction Library 2.1

SealSoft has released version 2.1 of its freeware xFunction library. This release includes a bug fix for the xFindFile function.
Posted Tuesday, February 24, 2004

Firebird 1.5 Final

The final release of Firebird 1.5 is now available for download from SourceForge.
Posted Tuesday, February 24, 2004

Clarion Training In Gauteng Cancelled

Raymond Dummer reports that the Clarion Training has been cancelled for Gauteng.
Posted Tuesday, February 24, 2004

Outlook-Related Products

SoftMasters has a new line of products designed to access information in Microsoft Outlook. There is one product for each specific Outlook item type: Outlook Calendar Manager; Outlook Contacts Manager; Outlook Journal Manager; Outlook Notes Manager; and Outlook Tasks Manager. Additionally, there are several templates included in each one of these five products, and each template was designed for a specific purpose. For example, the Outlook Calendar Manager product includes the following templates: Process Calendar, Calendar2Queue, Calendar Lookup Controls, Browse Calendar, Browse Calendar SelectButton and Browse Calendar Procedure, which allow you to quickly create a browse of Outlook appointments. The Outlook Contacts Manager, Outlook Journal Manager, Outlook Notes Manager and

Outlook Tasks Manager products includes similar templates to achieve the same tasks with Outlook contacts, journal items, notes and tasks, respectively. Each one of these products costs $25, or get the bundle for $75.
Posted Tuesday, February 24, 2004

New Clarion Wallpaper

A new Clarion wallpaper is now available for download from Gitano Software.
Posted Tuesday, February 24, 2004

Card Source Demo Updated

An updated demo of the card source application, with a fully working Crazy 8s game, is now available. Link and password for the source will be emailed to existing customers.
Posted Tuesday, February 17, 2004

RPM Drops Legacy Support

Due to poor response to the legacy add-on for RPM, Lodestar Software will not continue development on this product. The current legacy add-on for RPM will continue to be available but future legacy support for RPM and AFE has ceased as of this date.
Posted Tuesday, February 17, 2004

PayWire Credit Card Templates

PayWire has announced the availability of their Credit Card Templates. These templates, and the DLL, are available free of charge, and allow your customers to process MasterCard, Visa, Discover, and AMEX cards using your software. Also new is a revenue sharing program with developers of applications needing merchant credit card processing. For more information see the web site, or call 830.232.4241.
Posted Tuesday, February 17, 2004

SoggyNotes 1.2

SoggyNotes 1.2 is now available. New in Version 1.2: Add the data in the message body; Exclude the data attachment; Improved the delete message settings. Version 1.1 added emailing capabilities. All the Soggy templates are now listed at www.clarionshop.com.
Posted Tuesday, February 17, 2004

Firebird 1.5 RC 9

Kelvin Chua reports that Firebird 1.5 RC 9 is now available.
Posted Tuesday, February 17, 2004

Lindersoft Office Closed Until Feb 16th

The Lindersoft office will be closed for vacation from February 11th until 16th with no access to email or voicemail. Orders will still be accepted at the web site, and if you purchase a new license, you will receive your registration codes immediately. Please note that it is not be possible to send out registration codes for updates or answer customer service email until after the 16th.
Posted Wednesday, February 11, 2004

BLAT 1.2

BigLegacyToABCTamer Template, or BLAT version 1.2 is now available. This release includes bug fixes, and also has removed the Automatic Export feature as it is still unreliable in the IDE.
Posted Wednesday, February 11, 2004

Firebird Article

From Arno Rog comes this link to an article in Linux Journal, about the open source Firebird SQL database.
Posted Wednesday, February 11, 2004

etc2004 Web Site Update

Check out the updated pages for etc2004. Lots of good information to
Posted Wednesday, February 11, 2004

Fenix 1.5 Adds Mobile Support

Fenix 1.5 adds the ability to generate either standard .NET controls, or controls for the Mobile Framework. Just change a single template prompt, and Fenix will generate the ASPX pages with Mobile controls, so the .NET framework will determine the device capabilities and will render the best HTML (or WAP) for the target device.
Posted Wednesday, February 11, 2004

DevCon 2004 At DisneyWorld In September

DevCon returns! This three day conference will be at DisneyWorld in Florida, Monday September 20-Wednesday September 22. The venue is Disney's Contemporary Resort. A welcome reception will be held on Sunday evening, September 19. Clarion training will be offered pre- and post conference. Watch the SoftVelocity web site for further news.
Posted Wednesday, February 11, 2004

SoggyNotes

Soggy Systems has just released a new tool called SoggyNotes. Features include: Bug reporting and unlimited notes on a record; Support two types: Record notes and Support notes; Clients can enter multiple notes per record; Clients can enter support notes for the developer or developers can leave notes on changes made to the system; Full search on all notes. Template features include: Global template to set the procedures and hot keys; Three control sets, a list control with update controls for notes, an update control template, and search control template.
Posted Wednesday, February 11, 2004

BST 2.7

BST 2.7 contains a bug fix, more internationalization, and colors and reports for wall calendars.
Posted Wednesday, February 11, 2004

SoggyVersion

This version template works in conjunction with the Softvelocity Version template for Clarion 6. Verify version and allow your client to download the latest version if not up-to-date.
Posted Wednesday, February 11, 2004

New API Database & Template

The Clarion API Database had a completely rewritten APP and template. Versions available for C5.5 and C6.
Posted Wednesday, February 11, 2004

MSDE Usage Conditions Updated

Mark Riffey points out that there have been some positive changes to the conditions under which MSDE may be used.
Posted Wednesday, February 11, 2004

SoggyRunner

Soggy Systems has just released a new tool called SoggyRunner. Features include: Load the latest version of your application on the server without asking your users to close there application; Support for files include: Don't copy, Always replace, Replace when older and Never replace; Copy files to different directories; Set a file to launch with parameters. Freeware, commercial and source code versions available.
Posted Wednesday, February 11, 2004

Clarion Jobs Page

The Clarion Jobs Page was updated as well as the World Network of experienced Clarion programmers and companies with Clarion technology.
Posted Wednesday, February 11, 2004

ImageMan Templates Compatible With C6

Version 1.09 of the ImageMan Templates is compatible with both Clarion 5.5 and 6.
Posted Wednesday, February 11, 2004

Imaging Templates Compatible With C6

Version 1.23 of the Imaging Templates is compatible with both Clarion 6.5 and 6.
Posted Wednesday, February 11, 2004

1st Shareware Business For Sale

Gitano Software is selling 1st Shareware, an established shareware business. Assets include: Company name; Domain name (paid until 2008); Logo; Web site; Accent product with source; PowerSearch product with source; PDF Manager product with source; Source licenses for gCal, gCalc, gFileFind, gSec, gNotes, and a usage license for gReg License. The source code for gCal, gCalc, gFileFind,gSec, and gNotes cannot be resold individually, they need to be compiled as part of the included applications. For additional information see www.1stShareware.com.
Posted Tuesday, February 03, 2004

Imaging Templates 1.23 (C6 Compatible)

NextAge has released version 1.23 of the Imaging Templates. This version is compatible with both Clarion version 5.5 and version 6.
Posted Tuesday, February 03, 2004

Replicate 1.34 Beta

Replicate 1.34 beta is now available. Changes since 1.3 include: MsSQL work around (for bug in the driver that does not pass the REGET command through the callback); Fixed the RestoreBackup method; Support for ALIASes introduced; Supports FM3 and FM2 upgrading; Prepare methods for Online transport. The gold release, initially planned for January, has been postponed. Geoff will be on leave the last two weeks of February, and during that time support will be, according to Geoff, "a bit thin."
Posted Tuesday, February 03, 2004

ThinkData Releases xmlFUSE 2.1

ThinkData has released xmlFUSE 2.1 which includes support for the GeoMonster SOAP web services (see http://www.geomonster.com). These free web services allow you to verify United Kingdom, Canadian, and United States postal codes including the retrieval of the City, County, and Country. In addition, you can use GeoMonster to validate SMTP email address mailboxes and locate your users via their IP addresses. All of this has been added on the "MS SOAP Examples" demo form in xmlFUSE.
Posted Tuesday, February 03, 2004

ThinkData Sale Extended

ThinkData has extended its 25% off sale of all products through February 6, 2004.
Posted Tuesday, February 03, 2004

ABC Free Templates and Tools Updated (C6 only)

Changes to the ABC Free Templates and Tools include: Global Enter=Tab template, added workaround for C6 Gold; Fixed PRESSKEY(ShiftEnter) ("sticky shift key") bug; Changed browse to append SysID to sort orders (Global) template so that each element of the primary key is appended to the order separately.
Posted Tuesday, February 03, 2004

SoggyQueue Update

There is a new demo available for the SoggyQueue templates, beta 1.3. New features include: Tag records and process them; Tag all, Untag all and Swap tag support; Filter support for hand coded loads; First level filter for runtime filters; Faster loading of queue; Faster filtering of queue; Faster record locating; Bug fix to hand coded load.
Posted Tuesday, February 03, 2004

Cards Source Code For Sale

Able Software New Zealand is beginning to release a number of source code packages, including Point of Sale, Accounting, Horse Stud, Service Schedules, Fish Processing, Outdoor Equipment Sales, and more. To test sales and support procedures, Able is first releasing some source code that shows how to use the standard Windows cards.dll in your own games. $35.00 gets you the source for the demo application plus some notes on using the Windows cards.dll.
Posted Tuesday, February 03, 2004

Lodestar/Clarion Add-ons Web Site Redesign

Lee White's redesign of the Lodestar Software/Clarion Add-ons site has gone live.

Posted Tuesday, February 03, 2004

Gitano Software Specials

This week at Gitano Software: gReg half price sale, now $149.50; gFileFile sale, $50 off; Developer's Package, save $99, now $375.
Posted Tuesday, February 03, 2004

Free Words & Lines Counter

Ville Vahtera has provided a free utility to count the number of words and lines in your Clarion source (or any other text files). The program is about 32K in size.
Posted Tuesday, February 03, 2004

Clarion Magazine

Auto-Complete Files And URLs With SHAutoComplete

by Carl Barnes

Published 2004-02-26

One of the nifty things about Internet Explorer (and Windows Explorer) is the auto complete feature in the address bar. You'll also see this in the Common File Dialog. As you type URLs, or file paths, a list window appears with suggested URLs or files/folders. The function `SHAutoComplete` (which stands for Windows Shell Auto Complete) was added to Internet Explorer in version 5, and the good news is you can use it with Clarion, as shown in Figure 1. This makes it is *very* easy to add auto complete for files and URLs to any program.

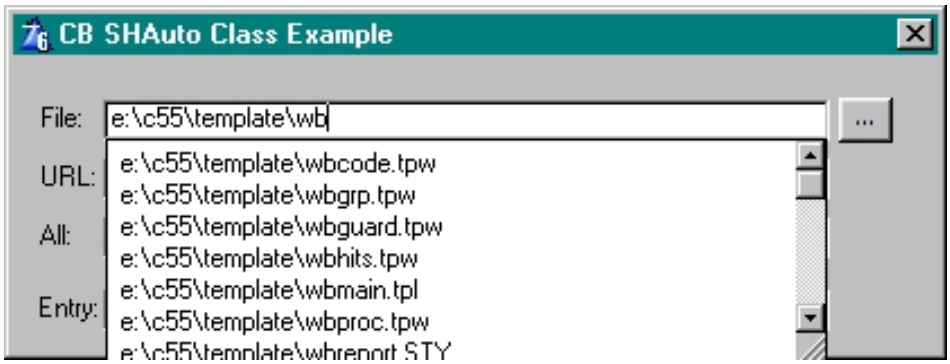


Figure 1. Using SHAutoComplete in a Clarion program

The main code change required to implement shell auto complete is a call to the `SHAutoComplete` API function for each control. If you read about `SHAutoComplete` on MSDN you'll find it has this prototype:

```
HRESULT SHAutoComplete(HWND hwndEdit, DWORD dwFlags);
```

All of the data types above are Clarion `LONG` types. Recently I've tended to leave API prototypes in their original Windows API data types and use equates to make the compiler use the correct Clarion data types. I think this has several advantages. The main one is it means fewer changes to the code. That means less time to convert and less potential for errors.

Also, with 64-bit Windows on the horizon it is possible that some of types that are currently 32-bit numbers will change to 64-bit, and some won't. Leaving prototypes in Windows API types will require only a single equate change if a type changes to 64-bit. If you change them all to `LONG`s you may have a lot of code review to do when porting.

Converting the prototype to Clarion syntax and providing the compiler equates for the API data types requires

the following code:

```
DWORD           EQUATE(LONG)
HRESULT         EQUATE(LONG)
HWND            EQUATE(UNSIGNED)
```

And here's the Clarion prototype:

```
SHAutoComplete(HWND hwndEdit,DWORD dwFlags),HRESULT,PASCAL
```

The first parameter `hwndEdit` is described as "Window handle of a system edit control". An edit control in Clarion is an `ENTRY`. In Clarion the handle to an entry is obtained using `?EntryEquate{PROP:Handle}` (substituting the equate of your own entry control for `?EntryEquate`).

If you try an `ENTRY` with `SHAutoComplete` you'll find it does not work correctly. It appears to work, but when the user presses Tab the selected item does not get placed into the control's `USE` variable. I did not try to debug why this did not work, so it is possible it can be fixed. I know from using Visual Test that Clarion `ENTRY` controls are custom and not native Windows edit controls. A `COMBO` control has the same issues.

The Clarion `TEXT` control is very close to a true Windows edit control and works perfectly with `SHAutoComplete`. If you add the `SINGLE` attribute then it will behave like an `ENTRY` and not allow multiple lines. The `ENTRY` control has a limit of 255 characters. A path and file name can be 260 characters, and a URL can be even longer. A `TEXT` control does not have the 255 byte limitation so is better suited to the task of file/URL entry. With a single call of `SHAutoComplete(?TextControl{Prop:Handle},0)` the auto complete will be enabled with default parameters.

I have verified this works correctly under Clarion 5.5 and 6. It did not work quite the same under Clarion 5. The tab key would not pick the item from the list and move to the next control. Instead, tab would move between selections in the list. To pick an item from the list the user has to press enter. This sounds reasonable, but then the user cannot tab out of the control and must use the mouse. I did not try too hard to "fix" C5, but again, it's possible this can be made to work.

SHAutoComplete flags for fill data

While you can call `SHAutoComplete` with the second parameter flags set to zero, and get very good default behavior, there are about a dozen flag equates that can provide more exact control over what is contained in the list and how it functions. There are a few "gotcha's" in using the flag equates which I will explain in a moment. In the download I have included an example project named "SHAC Explorer". It will let you try almost every possible combination. Figure 2 is a screen shot showing the tab that lets you try some of those equates:

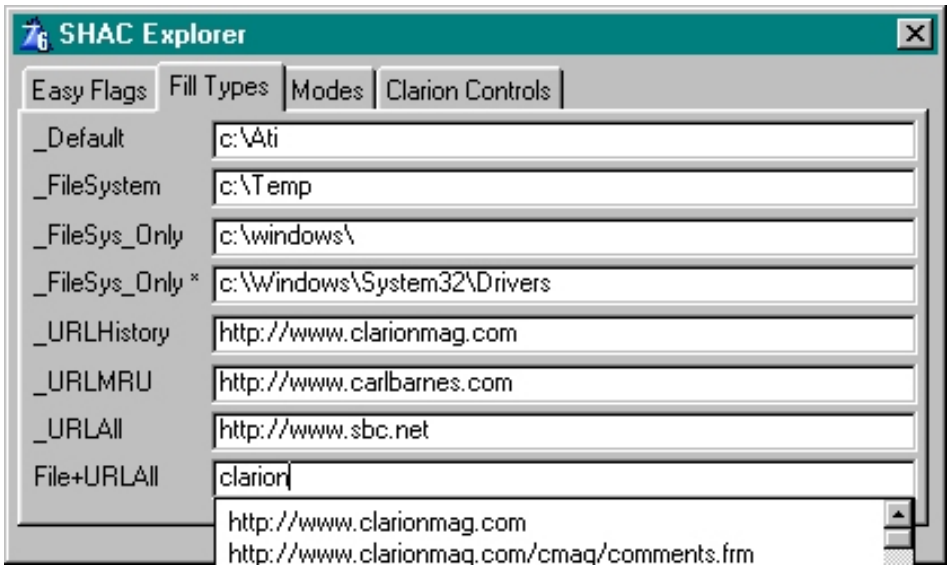


Figure 2. Exploring fill types

The first set of flags to consider are those that specify what type of data will be in the fill list: files, URLs or both. I think if a screen entry were requesting a file it would confuse a user to have URLs showing in the fill list. The fill data equates are listed below:

```
SHACF_DEFAULT EQUATE(00h)
```

This is the zero flag I used above for the default configuration. The fill list will contain files and URLs (equivalent to using `SHACF_FILESYSTEM+SHACF_URLALL`). *This is a "special" flag and cannot combined (OR'd) with any other flags.*

```
SHACF_FILESYSTEM EQUATE(01h)
```

Include the file system in the fill list (e.g. `c:\C6\Bin\.`) as well as shell virtual folders such as Desktop or Control Panel. In my testing I was not able get any virtual folders to appear in the list no matter what equate I used.

```
SHACF_FILESYS_ONLY EQUATE(10h)
```

Include the file system in the fill list, but do not include shell virtual folders. In my tests when this equate is used alone *there is no fill list generated*, i.e. it doesn't work. It must be combined with `SHACF_FILESYSTEM` e.g. `SHACF_FILESYSTEM+ SHACF_FILESYS_ONLY`. I found no mention of this on MSDN or any place else. Also note above that I couldn't get the `SHACF_FILESYSTEM` equate to show virtual folders anyway. So enough about that…

```
SHACF_URLHISTORY EQUATE(02h)
```

Include URLs from the User's History in the fill list.

```
SHACF_URLMRU EQUATE(04h)
```

Include URLs from the User's Recently Used List in the fill list.

```
SHACF_URLALL EQUATE(SHACF_URLHISTORY+SHACF_URLMRU)
```

Include URLs from both History and Recently Used.

SHAutoComplete flags for fill modes

Auto completion has two modes it can use to display the potential completion value to the user: suggest mode and append mode You are probably most used to seeing "Suggest mode" where a list of suggestions opens in a scrollable list window below the edit control, as in Figure 3.

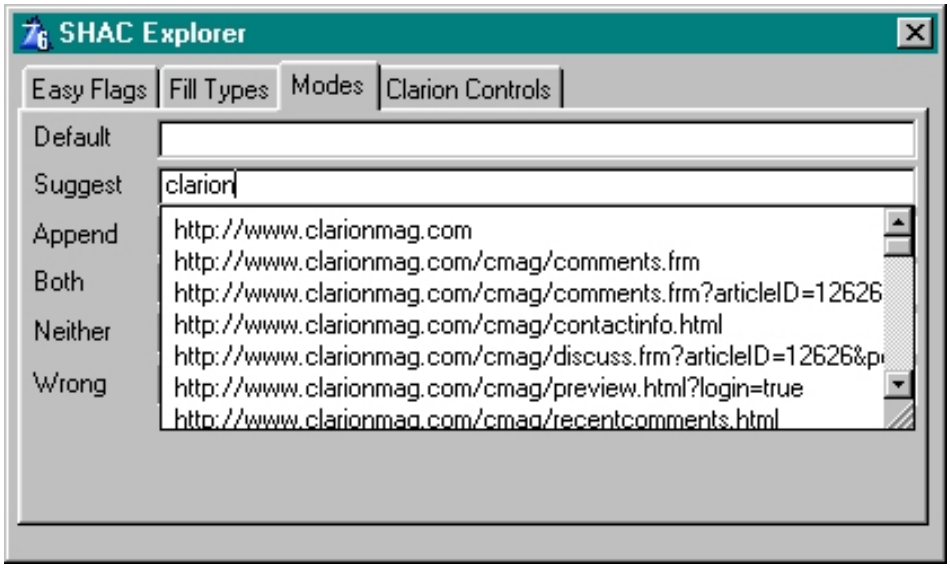


Figure 3. Suggest mode

In append mode the completion value is shown in the edit control itself to the right of the caret, i.e. it is appended to the user's partial entry, and shows as selected or highlighted, as in Figure 4. I don't care for this mode since it only shows a single value. The user can still use the arrows to scroll up and down through the list of values, but this is not obvious enough.

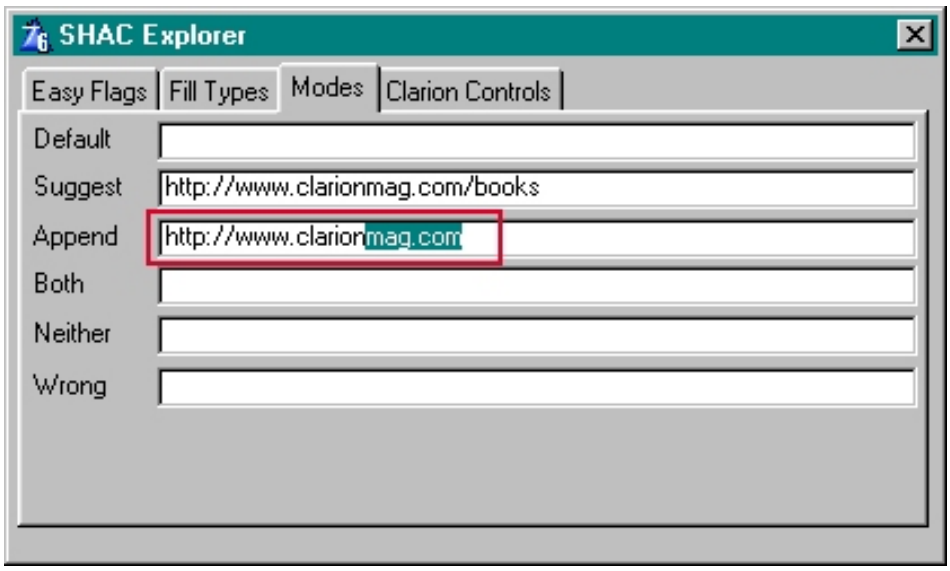


Figure 4. Append mode

The two modes are not mutually exclusive; you can turn on both of them. This has pros and cons. I think it would be somewhat confusing for some users to see both options, as in Figure 5.

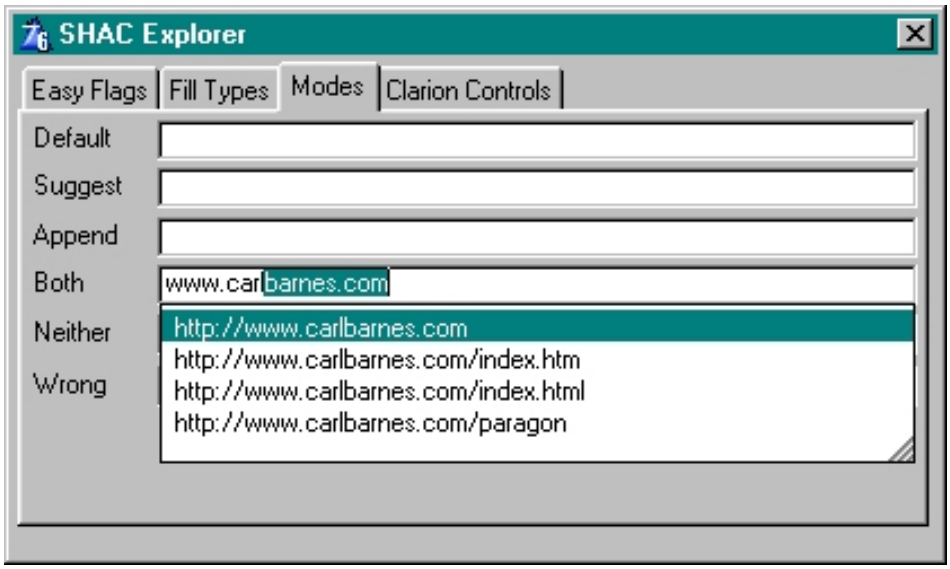


Figure 5. Append and suggest mode combined

According to MSDN the default mode can be set by the user in Internet Explorer on the Options menu under Tools on the Advanced tab. The selections chosen are stored in the registry under this key:

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\AutoComplete
```

I was not able to see changes I made in IE affect the way my Clarion program worked. It seemed to always use the suggest list. The equates for mode are:

```
SHACF_AUTOSUGGEST_FORCE_ON EQUATE(10000000h)
```

Ignore the registry value and force the autosuggest mode on. A selection of possible completed strings will be displayed as a drop-down list, below the edit box.

```
SHACF_AUTOSUGGEST_FORCE_OFF EQUATE(20000000h)
```

Ignore the registry default and force the autosuggest mode off.

```
SHACF_AUTOAPPEND_FORCE_ON EQUATE(40000000h)
```

Ignore the registry value and force the autoappend mode on.

```
SHACF_AUTOAPPEND_FORCE_OFF EQUATE(80000000h)
```

Ignore the registry default and force the autoappend mode off.

I found these equates somewhat confusing at first. You specify none of them if you want the default settings. I would probably explicitly specify the mode(s) I wanted so my program would run in a predictable way on all systems. To correctly override the default you must specify two equates, one AUTOSUGGEST to turn it on/off, and one AUTOAPPEND to turn it on/off. My preferred method would be to use the "Suggest mode" only. The

mode equates to use are SHACF_AUTOSUGGEST_FORCE_ON + SHACF_AUTOAPPEND_FORCE_OFF + … .

One last detail about mode equates; if you specify any SHACF_AUTOxxx equates you must specify a SHACF_FILE or SHACF_URL equate. *You cannot use SHACF_DETAIL with SHACF_AUTOxxx*. This is demonstrated in SHAC Explorer on the Mode tab with the entry labeled "Wrong". No matter what you type an auto completion suggestion will not appear. The class supplied in the download will assert if you try to use an improper flags combination.

SHAutoComplete flags simplified

If the above is just too much equate information, and it probably is, I have declared three "Easy Input" equates that are all you'll need to use when calling SHAutoComplete. You simply have to decide if you want the fill list to contain files, URLs or both. There is also a fourth equate to define the desired mode, which I defined for Suggest mode only. My equates are:

```
SHACR_MODE EQUATE(SHACF_AUTOSUGGEST_FORCE_ON + SHACF_AUTOAPPEND_FORCE_OFF)
```

This is the mode to use in the following equates. My choice turns the suggest drop list on and append mode off.

Fill list will contain files without virtual folders:

```
SHACR_INPUT_FILE EQUATE(SHACR_MODE + SHACF_FILESYSTEM + SHACF_FILESYS_ONLY)
```

Fill list will contain URLs (history and recently used):

```
SHACR_INPUT_URL EQUATE(SHACR_MODE + SHACF_URLALL)
```

Fill list will contain both URLs and files:

```
SHACR_INPUT_ALL EQUATE(SHACR_MODE + SHACF_URLALL + SHACF_FILESYSTEM +
SHACF_FILESYS_ONLY)
```

You can try these three equates in SHAC Explorer on the first tab named Easy Flags, as shown in Figure 6.

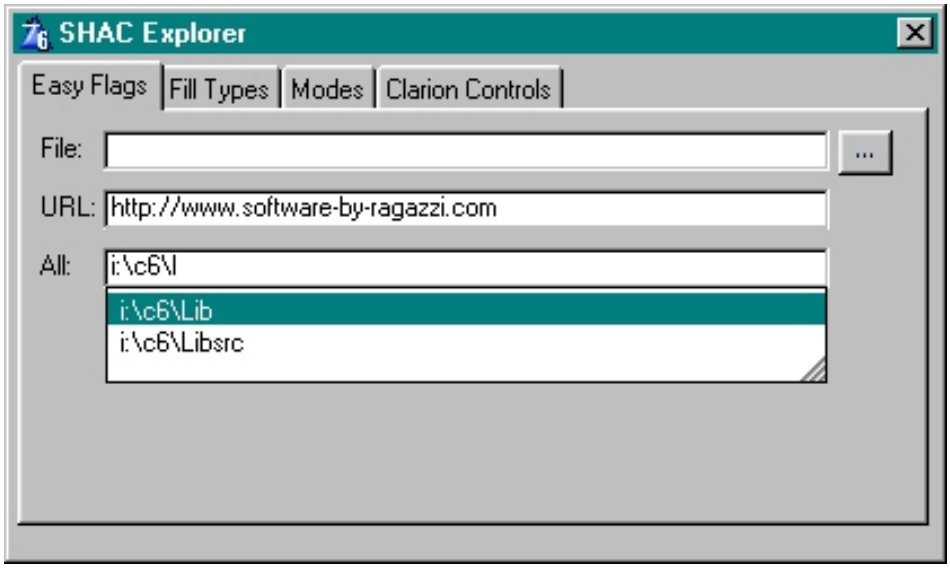


Figure 6. Exploring the Easy Flags

COM library requirements

To call `SHAutoComplete` requires that you first call `CoInitilize` to get the COM library setup. MSDN says that you must call `CoInitilize()` once on each thread. I have included a link below to MSDN for complete information. Don't forget to call `CoUninitialize()` on your way out. You can call `CoInitilize` more then once on each thread, but each call must be paired with a call to CoUninitialize. More on that later.

Return values

`SHAutoComplete` returns an `HResult`, which is the normal type of return value from COM functions, and is a signed 32-bit number. If the function succeeds it returns `S_OK EQUATE(0)`. If it fails it returns a negative number with the error information. The most common reason you'll get an error is because you failed to call `CoInitialize`. In my testing this results in a return of `-2147024882` or `8007000Eh`.

SHAutoComplete link and run requirements

`SHAutoComplete` is defined in the "Shell Lightweight API" (file name shlwapi.dll), which started shipping with IE 5. To build your program the linker will require a lib file for this DLL.With the example I have included a shlwapi.lib library file which I made using LibMaker. If you link your EXE using this lib your program will require shlwapi.dll to be present on the end user's system. For any user that does not have IE5, your program will fail at load time with a missing DLL error and not run at all.

If you choose to require your user to have IE5 there are many useful functions in the Shell Lightweight DLL that handle strings, paths, URLs, registry entries and a few other things. I have included a link below to the MSDN info on all the functions available. I think it is reasonable to require IE5, but you'll want to carefully consider that choice.

To ensure that your program will never fail to load because of a problem with shlwapi.dll, you need to load the DLL at runtime. This is fully described in Larry Sand's excellent CMag article "Loading DLLs at runtime". If you use this approach, and the DLL fails to load, your program will run just fine, only the auto complete feature will not work.

An ABC Class to make it easy

Included with the download is an ABC compliant class I named `SHAutoCompleteClass` that implements everything I have learned about `SHAutoComplete`, and makes it dirt simple to use. For each control you just call one of three functions based on the type of stuff you want in the fill list. The design uses Larry's `LoadLibClass` to load the swlwapi.dll at runtime and gracefully handle things if the DLL does not exist. (I did not actually test this failing with IE4, but I did simulate it; caveat emptor.)

You can add shell auto complete to your program with just the below lines of code:

In the global or module includes embed:

```
INCLUDE('CBSHAuto.inc')
```

Data declaration:

```
SHAuto SHAutoCompleteClass
```

Embed code (use appropriate text field equates)

```
SHAuto.Input_FILE(?EasyInpFile) !Files in List
SHAuto.Input_URL (?EasyInpURL)  !URLs in List
SHAuto.Input_ALL (?EasyInpAll)  !Files and URLs in list
```

Summary

This wraps up how to use `SHAutoComplete` in Clarion. There is one limitation brought to my attention by Jim Kane: the user must tab out of the control. Jim says he tried everything to allow pressing enter, similar to the way IE works, but was unable to do it. If you are trying to make your own browser that might be annoying, but for data entry of file names and URLs I can live with it.

With a little preparation and single API call per control you can have this nifty feature in your program that will aid your user in entering files or URLs. Remember that it only works correctly with `TEXT` controls. Be sure and download the SHAC Explorer below and play around with the various options this function offers.

Download the source

Download the source for Larry Sand's DLL Loader

Related Links:

- MSDN Using Autocomplete
- MSDN IautoComplete
- MSDN HResult
- Coinitialize
- Shell Lightweight Utility Functions
- Loading DLLs At Runtime by Larry Sand (Clarion Magazine)

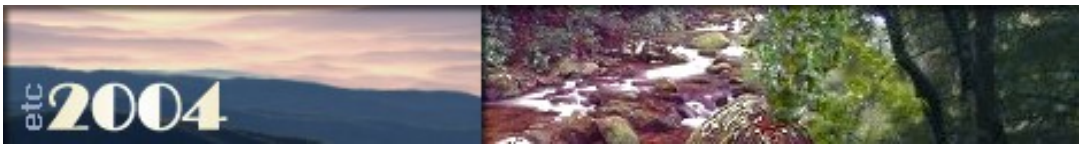
Carl Barnes is an independent consultant working in the Chicago area. He has been using Clarion since 1990, is a member of Team TopSpeed and a TopSpeed Certified Support Professional. He is the author of the Clarion utilities CW Assistant and Clarion Source Search.

Reader Comments

Add a comment

Great article, Carl. I can't wait to add this feature to...
My bad - Carl had sent me the correct zip and I attached...

Clarion Magazine



Template Styles in Clarion 6

by Russell Eggen

Published 2004-02-27

January 2003 is when we got to see the first version of Clarion 6 in the Early Access program. There were a lot of changes to the templates, but there were also some enhancements to the template language which few noticed.

Want a clue? Look at Figure 1.

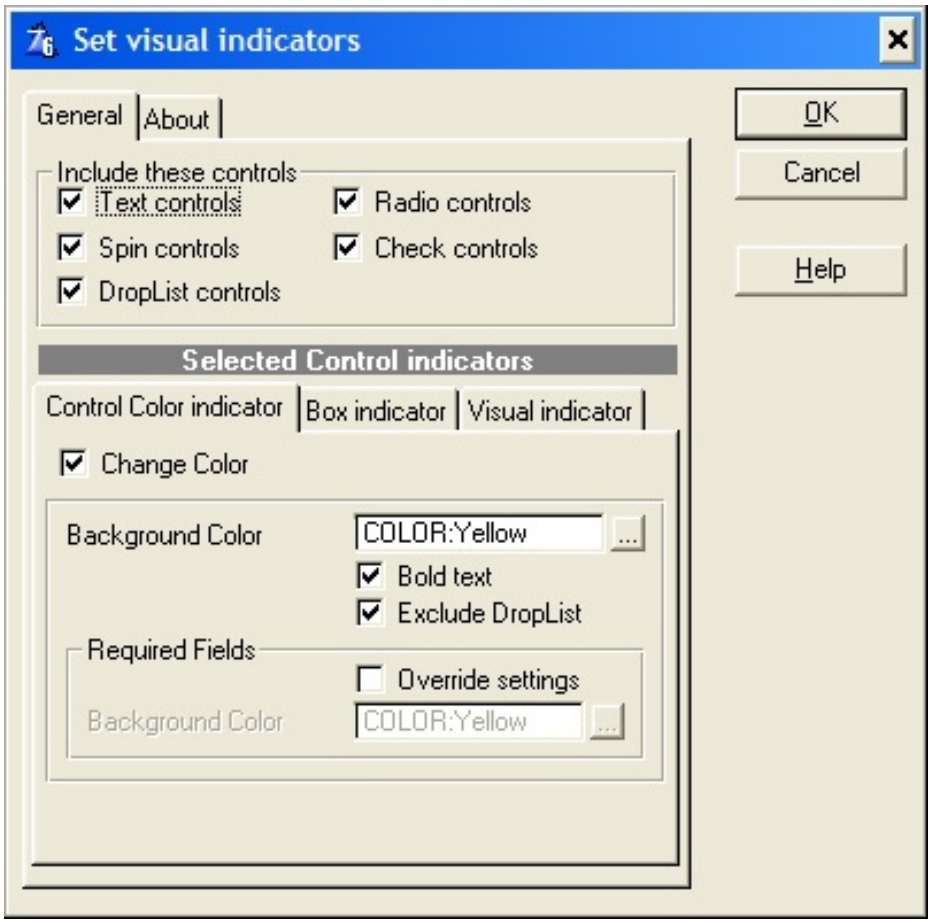


Figure 1. Set Visual Indicators dialog.

The background color of the "Select Control indicators" text in that image jumped out at me. If you look closer, you can see that the foreground color is different too. This is interesting as it could open the door to

all sorts of template dialog styles. No more boring template dialogs!

So how was this done? A search of the template files reveals this line in ENHANCED.TPW, in the `%EnhanceFocusGlobalPrompts` group:

```
#DISPLAY('Selected Control indicators'), AT(5,,185),
   PROP(PROP:FontColor, 0FFFFFFH),PROP(PROP:Color,0808080H),
   PROP(PROP:FontStyle,700),PROP(PROP:Center,%True
```

The `PROP()` part of the statement appears new. Of course, the help is vastly improved in Clarion 6 as well. Searching the help under `#DISPLAY` (since help is organized by statements and commands), you find this item:

> PROP Specifies a property to assign to the prompt text. Name designates the property name equate (Example: PROP:FontColor) and value is the value assigned to the named property (Example: 0FFFFFFH)

In addition, there is a nice bonus in the help; a link to another template statement that also uses `PROP()`, specifically `#PROMPT`.

I'm thinking now of the future third party templates that are easy on the eyes. Of course, this could also be a nightmare. Who wants their favorite template in the old Hot Dog color scheme?

But the point is that any displays you wish to present to a developer may now take on additional characteristics, such as font and color. More than that actually, any property that you may assign to a string in your Clarion applications, you may now assign to a template string or prompt.

Template prompts are where you may assign the type of control to use, entry, check, list box, etc. This means that attributes that are applicable with those controls may also appear in template dialogs.

An example, please

Let me first demonstrate with a template that does not do anything - it's just cosmetic showing what you can do with the `PROP()` attribute. To keep it simple, I'll only use the `#DISPLAY` statement. Here is the entire template (NOTE: line breaks added for formatting purposes - use the downloadable source instead):

```
#TEMPLATE(NewStuff,'Demo of new attributes'),FAMILY('ABC'),FAMILY('CW20')
#SYSTEM
  #TAB('New Stuff')
    #BOXED(' What is new in ' & %CWVersion & '? ')
      #DISPLAY
      #DISPLAY('NOTE: This is not a working template.')
               ,PROP(PROP:FontColor, 02626FFH),PROP(PROP:FontStyle,700)
      #DISPLAY('If this had been an actual working')
               ,PROP(PROP:FontColor, 02626FFH),PROP(PROP:FontStyle,700)
      #DISPLAY('template, you would have been shown ')
               ,PROP(PROP:FontColor, 02626FFH),PROP(PROP:FontStyle,700)
      #DISPLAY(' instructions on its use.')
               ,PROP(PROP:FontColor, 02626FFH),PROP(PROP:FontStyle,700)
```

```
    #DISPLAY
  #ENDBOXED
#ENDTAB
```

Register the template and then double-click on the new template in the registry. This is what you see:

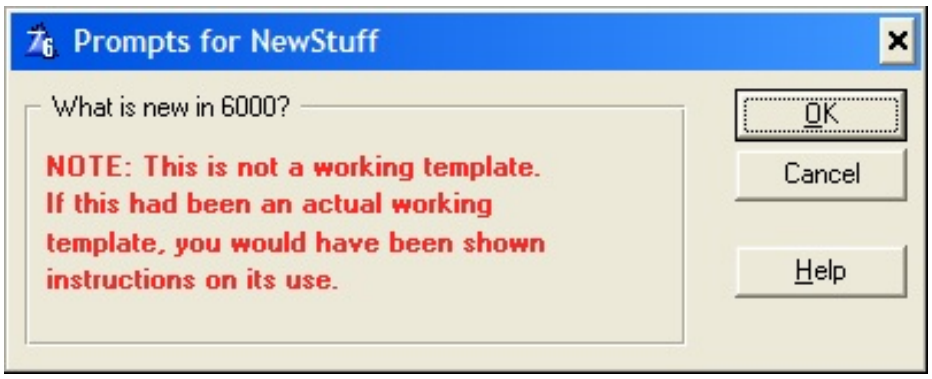


Figure 2. An example of color change.

The reason I enclosed the prompts in the #SYSTEM statement is so double clicking on the template registry shows the above figure. It also means it appears in the IDE, under the Setup|Application options menu. You'll need to scroll all the way to the right to see it. This is a good way to test without requiring an application.

Backward compatibility

Colorful prompts are nice, but what about backward compatibility? The obvious problem is that PROP() does not exist in prior versions. I don't know of any vendors who want to maintain two complete versions of a given set of templates, especially if the only difference is cosmetic.

You can create version-specific template features with the built-in symbol %CWVersion. The value in that symbol for Clarion 6 is 6000 - just use the #IF statement to test the value. The only drawback is that you cannot use #IF inside a #BOXED statement. Remember, this is the template language, and it has some oddities, to say the least. However, you are not stuck, as #BOXED has a built-in #IF statement equivalent: the WHERE(<some condition>) attribute. You use a conditional #BOXED statement like this:

```
#BOXED(' What is new? '),WHERE(%CWVersion >= '6000')
```

In this example the #BOXED statement applies only if the Clarion version is 6000 or greater. In prior versions, it won't display at all. You'd want something like the following for earlier versions:

```
#BOXED(' What is new in ' & %CWVersion & '? '),WHERE(%CWVersion < '6000')
  #DISPLAY
  #DISPLAY('NOTE: This is not a working template.')
  #DISPLAY('If this had been an actual working')
  #DISPLAY('template, you would have been sent ')
  #DISPLAY(' instructions on its use.')
  #DISPLAY
```

```
#ENDBOXED
```

Notice the `PROP()` attributes are not there. In Clarion 5.5H, you get the dialog shown in Figure 3.

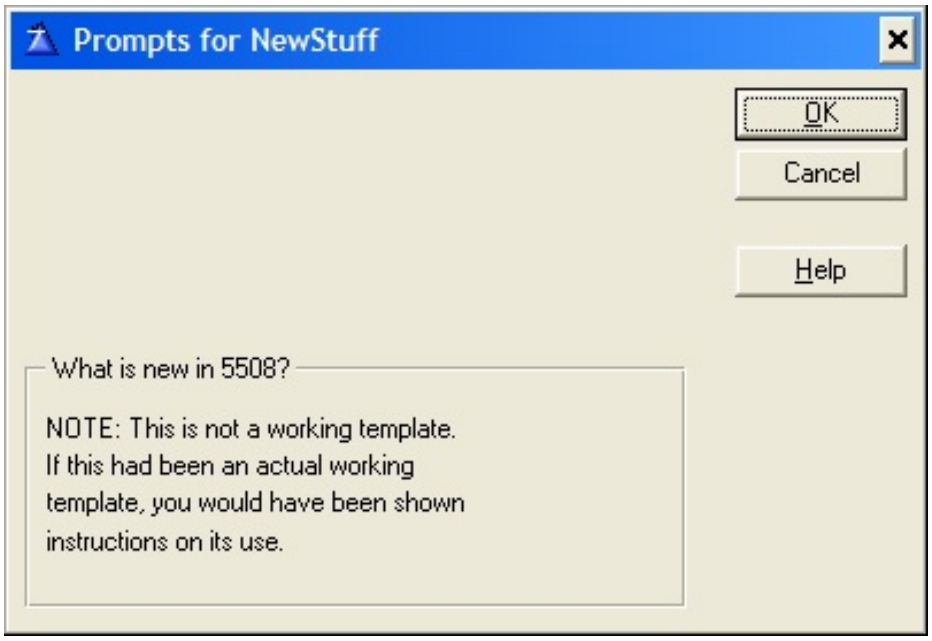


Figure 3. 5.5H version shown.

OK, the `WHERE(<some condition>)` statement works. Why is the text shoved down the dialog? What happens if you look at the template in C6, after adding the pre-C6 code? You'll see something like Figure 4.

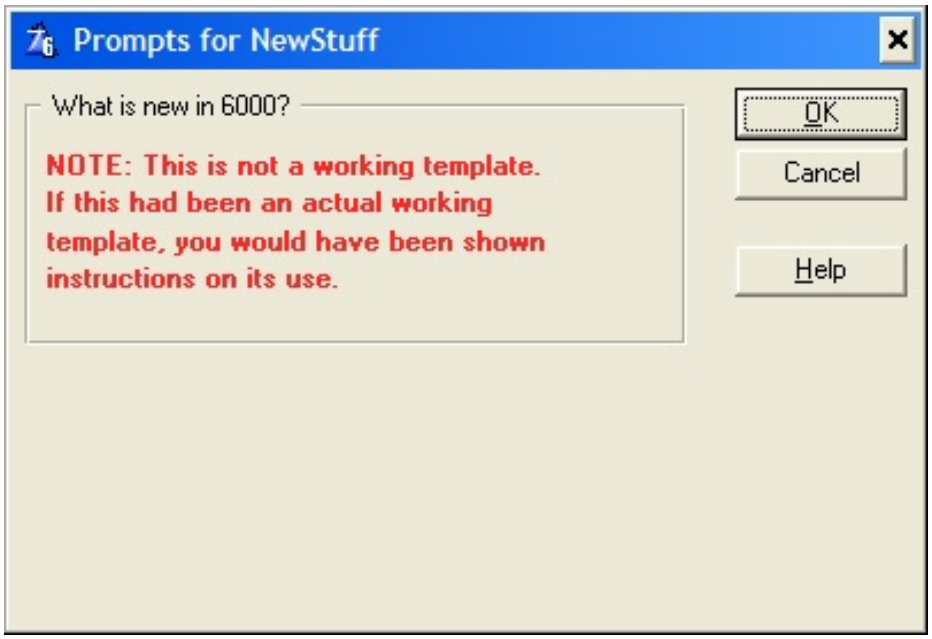


Figure 4. Clarion 6 with extra space at the bottom.

Fix one problem, create another

Well, this is not good as the dialogs from both versions now look silly and unprofessional. The problem is that both `#BOXED` blocks show, one without any text as the text is hidden. Which one is hidden depends on

the Clarion version you use to view the dialog.

To fix the problem, add SECTION and AT attributes to the #BOXED statement so it appears like this (line break added):

```
#BOXED(' What is new in ' & %CWVersion & '? '),
  WHERE(%CWVersion <    '6000'),SECTION,AT(,5)
```

The SECTION means all AT() attributes for the prompts are positioned relative to the start of the #BOXED section. The AT() works exactly like the Clarion language AT statement. Therefore the two dialogs now appear as shown in Figure 5 and 6.

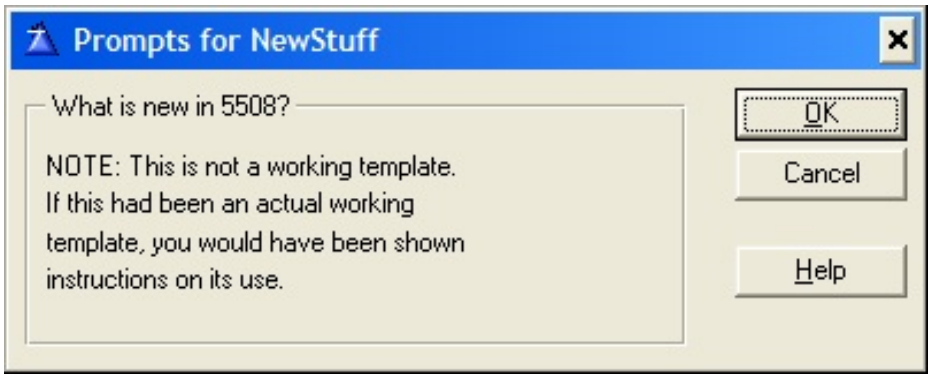


Figure 5. Fixed 5.5H dialog.

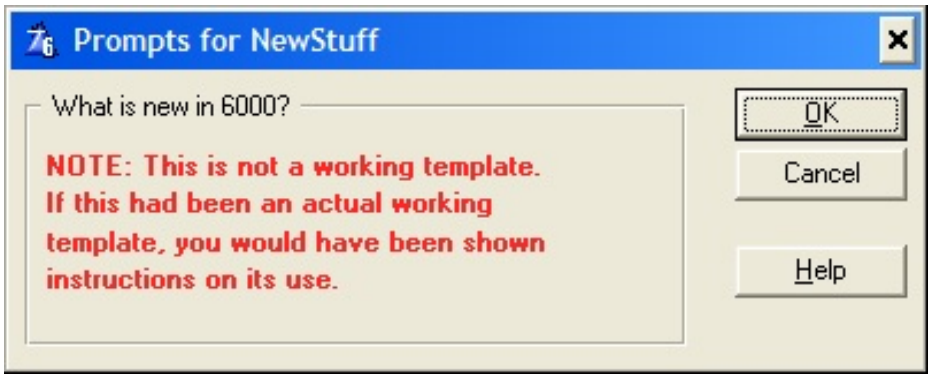


Figure 6. Fixed clarion 6 dialog.

What I've really done is overlay the two #BOXED statements. The WHERE() attribute and the condition I specified hides one or the other from the dialog. It also means that attributes that are not understood by previous versions of Clarion don't matter, as the filtering condition of the WHERE() attribute means the template compiler (registration process) skips over them, treating them as comments.

You may think of #BOXED and #IF, in conjunction with %CWVersion, as the OMIT and COMPILE statements for templates.

Summary

You can use Clarion 6 enhanced template commands and keep backward compatibility. You don't need two nearly identical sets of template files to do it. Best of all, using #BOXED and #IF is easy; you don't need to

put a lot of work into making sections of your templates backward compatible.

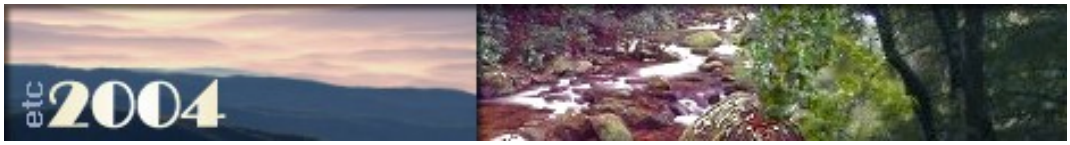
Download the source

Russ Eggen has been using Clarion since 1986. Until about 1996, he was using it for business applications, mostly accounting programs. Afterwards he joined Topspeed as a consultant, and later as an instructor. He was a founding member of SoftVelocity when that company formed from Topspeed in May 2000. He left SoftVelocity in January 2001 and now works for his own company, RadFusion Inc. He still teaches and lectures, and is currently working on a new book and setting up a local Clarion classroom. Russ enjoys flying, scuba, and applied philosophy, and with great effort you might coax him into political discussions.

Reader Comments

Add a comment

Clarion Magazine



Creating Derived ABC Classes With a Template

by Harley Jones

Published 2004-02-27

In this article, I'll try to explain a bit of the behind-the-scenes magic that Clarion performs on ABC windows and controls, and how you can do the same. I'll assume you have a bit of familiarity with template writing, and won't bore you with the mundane details.

One of the biggest mysteries to non-OOP, Clarion programmers is how the embed points work with a large class library made up of external files. In Legacy code, embed points are used to slip hand-code right in the middle of the template-generated code. If there are any external files, they are either data or completely standalone functions. In the ABC template set, the IDE lets developers embed code right in the middle of a class's method. But that class and its methods are stored in external CLW files. How does the IDE do it?

If no changes are made to the class (i.e., the developer does not embed any code), the templates simply generate an instance of the class:

```
ThisWindow WindowManager
```

But if the developer does embed code (or use a template that embeds code), the templates generate a derived instance of the class. It also generates a definition and declaration (with your embedded code) for each overridden method and any new properties or methods:

```
ThisWindow              CLASS(WindowManager)
Init                      PROCEDURE(),BYTE,PROC,DERIVED
NewProperty               LONG
NewMethod                 PROCEDURE(),BYTE
                        END
```

New properties and method can be added by using the Classes tab. (For the Window template, this tab is found by clicking on Window Behavior on the Procedure Properties screen. Just about every other template is found by clicking on Extensions.) See the online help for more information about using the Classes tab.

Without further ado, here is the template code I've written to add a property to the class:

```
#GROUP(%AddProperty, %pClassProperty, %pClassDataType), AUTO
  #DECLARE(%PropertyPresent)
  #DECLARE(%LastInstance)
```

```
  #SET(%PropertyPresent, %False)
  #SET(%LastInstance, 0)
  #FOR(%NewClassPropertyItems)
    #IF(%LastInstance < %NewClassPropertyItems)
      #SET(%LastInstance, %NewClassPropertyItems)
    #ENDIF
    #IF(UPPER(%NewClassProperty) = UPPER(%pClassProperty))
      #SET(%PropertyPresent, %True)
      #BREAK
    #ENDIF
  #ENDFOR
  #IF(%PropertyPresent = %False)
    #ADD(%NewClassPropertyItems, %LastInstance + 1)
    #SET(%NewClassProperty, %pClassProperty)
    #SET(%NewClassDataType, %pClassDataType)
  #ENDIF
  #SET(%DeriveFromBaseClass, %True)
```

Pretty simple, eh? Here's a brief explanation of what's going on. First off, the #GROUP assumes that %NewClassPropertyItems is in scope. This should be a safe assumption in the shipping ABC templates – if your templates use this GROUP, then it's up to you to be sure %NewClassPropertyItems is in scope. And what is %NewClassPropertyItems? It's a multi-valued symbol created by the ABC template set. It holds any properties created by using the Classes tab. The #GROUP then searches through the existing collection looking for the property that is to be added. This is done to ensure the property isn't added twice. If the property is *not* found, it is added to the collection and two of its dependent symbols are set, %NewClassProperty and %NewClassDataType. The last line tells the ABC template set that the current class has been derived, and to generate the code accordingly.

Here's the code to add a method. It is almost identical except for the name of the collection and its dependents:

```
#GROUP(%AddMethod, %pMethodName, %pMethodPrototype), AUTO
  #DECLARE(%MethodPresent)
  #DECLARE(%LastInstance)
  #SET(%MethodPresent, %False)
  #SET(%LastInstance, 0)
  #FOR(%NewMethods)
    #IF(%LastInstance < % NewMethods)
      #SET(%LastInstance, % NewMethods)
    #ENDIF
    #IF(UPPER(%NewMethodName) = UPPER(%pMethodName)
       AND UPPER(%NewMethodPrototype) = UPPER(%pMethodPrototype))
      #SET(%MethodPresent, %True)
      #BREAK
    #ENDIF
  #ENDFOR
  #IF(%MethodPresent = %False)
    #ADD(%NewMethods, %LastInstance + 1)
    #SET(%NewMethodName, %pMethodName)
    #SET(%NewMethodPrototype, %pMethodPrototype)
  #END
```

```
  #SET(%DeriveFromBaseClass, %True)
```

NOTE: These collections are auto-numbered, so `%LastInstance` is used to ensure new items do not trample other items in the collection. For example, `%LastInstance` will not necessarily be equal to `ITEMS(%NewClassPropertyItems)`.

These `#GROUP`s can be used with the following syntax:

```
#CALL(%AddProperty, 'PropertyName', 'PropertyDataType')
#CALL(%AddMethod, 'MethodName', 'MethodPrototype')
```

Now the coder (not the template writer), can use all the properties and methods of the class in the same fashion. He won't be able to tell that a method has been added by hand or by template. If the developer deletes one of the template-added methods or properties (using the Classes tab), the `#GROUP` will simply add it back when the APP is next generated.

As an example, let's say the template writer created a new method using the following line:

```
#CALL(%AddMethod, 'NewMethod', '(),BYTE')
```

To add actual content to a new method, the template writer can use the following syntax, for instance to add code to the Data Section (line break added at the WHERE clause):

```
#AT(%NewMethodDataSection, %ActiveTemplateParentInstance, %ClassItem),
   WHERE(%NewMethodName = 'NewMethod' AND %NewMethodPrototype = '(),BYTE')
bRetval              BYTE, AUTO
#ENDAT
```

Here's a template fragment to add code to the Code Section (line break added at the WHERE clause):

```
#AT(%NewMethodCodeSection, %ActiveTemplateParentInstance, %ClassItem),
  WHERE(%NewMethodName = 'NewMethod' AND %NewMethodPrototype = '(),BYTE')
bRetval = True
RETURN bRetval
#ENDAT
```

To add hand code, a developer would simply use the embeditor, the same as for any other class method.

And that's it. There is now a clean and easy interface for both the template writer and the developer. The two biggest benefits of this approach are the template completely hides the implementation of any new methods, and everything is seamlessly integrated with the IDE. This approach also allows developers to "extend" objects, instead of deriving them. For example, I've attached a sample `#CONTROL` template which provides a BrowseBox Refresh button . The approach shown here allowed me to extend the specific browse, without affecting the rest of the APP, and use particular template settings to generate different methods and/or properties. And because these new methods are used from a derived class, the developer has access to Protected methods which are usually unavailable. The developer gets exactly what she wants with no extra baggage.

If you read through the template, you'll also find a couple of `#GROUP`s which will remove methods and

properties. Currently, the Clarion Template language has no way of telling a template that it is being removed and needs to clean up any mess it has made. These two groups (`%RemoveProperty` and `%RemoveMethod`) are used to do exactly what they say. If and when SoftVelocity adds a method of telling a template it is being removed, these groups could help with the cleanup.

Download the source

Harley Jones graduated from a math and science school, so he could go to a liberal arts college and get an English degree, so he could become a programmer. He listens to loud music while he works, likes to eat gummi savers, and is always looking for ways to program more by programming less. He lives in Mobile, Alabama, where he spends all his money on his wife, daughter, and the occasional comic book.

Reader Comments

Add a comment