Clarion MAGAZINE

Search
Go

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

March 2001 PDF

This PDF contains all of the March 2001 articles. You need access rights to all the March 2001 issues to dowload this file. For a list of all available PDFs see the PDF download page.

Posted Wednesday, March 28, 2001

Dynamic Filters: The Theory Behind the Facts

Clarion's templates hide complexity. The ABC classes are supposed to also. But too often we developers re-introduce complexity.Once you realize that filters can be created in only two ways, it is really much easier to optimize filter expressions and performance. Part 1 of 3.

Posted Tuesday, March 27, 2001

The Clarion Advisor: Another Use of Loops

This edition of the Clarion Advisor is about the Clarion LOOP construct. Most developers use LOOP to iterate through the contents of a queue or table, or to perform some repetitive calculation. But LOOP can also simplify IF statement processing.

Posted Monday, March 26, 2001

Whitemarsh Project Management: Architecture And Concept Of Operations

This Whitemarsh paper discusses the importance of project management and explains the Whitemarsh project management environment. In PDF format only.

Posted Friday, March 23, 2001

Book Review: The Art & Science of Web Design

Web development means a lot more than slapping a few pages on a web server. If you have any interest in making your web

SysList 1.1 Released

VariView Adds Message Streaming

Open Clarion News Server

PickDBG Debugger Article

Free Auto Build Number Template

Clarion 5.5c Released

Keystone Reduces Pricing

Imaging Templates 1.10 Released

Web Site Of The Week

Buggy 2.1.1 Available

SoftVelocity FAQ System

Search Engine Profile Exchange Updated

VariView 2.00 Released

Linder SetupBuilder 3.51 Available

Free G-Notes For Clarion Editor

site a better experience for your users, read this book. Reviewed by Dave Harms.
Posted Tuesday, March 20, 2001

Web Splash Screens

It's common for a Windows application to need to open a browser window to display a web page. The easy way to do that is by passing a URL to ShellExecute(), but as Carl Barnes shows there are a few tricks you can use to make ShellExecute even more productive.
Posted Tuesday, March 20, 2001

The Clarion Challenge Results: Useless Tab Text

Several weeks ago I tried to plumb the meaning of inverted text on sheet tabs, and in the process came up with a small but useless application that sent tabs scurrying around all four sides of a sheet. I posted my code and challenged ClarionMag readers to fix a glaring bug in the code, write the code more compactly, and finally to come up with a real application for inverted tab text.
Posted Wednesday, March 14, 2001

Introduction To SQL: Part 3

In the third installment of this new series, Dave Harms looks at some of the differences between TPS tables and SQL tables.
Posted Wednesday, March 14, 2001

Feature Interview: SoftVelocity's Bob Zaunere

Bob Zaunere, SoftVelocity's President and CEO, recently spoke with Dave Harms about the Clarion product line and SoftVelocity's plans for the future.
Posted Monday, March 12, 2001

Introduction To SQL: Part 2

In the second installment of this new series, Dave Harms explains how to create an SQL database.
Posted Tuesday, March 06, 2001

The Cranky Programmer: To Patch or Not to Patch, That is the Question

Last week the long awaited first batch of fixes for Clarion 5.5 was actually unleashed upon the world. This week, Clarion Magazine unleashes Cranky upon

the unsuspecting patch.

Posted Tuesday, March 06, 2001

March 2001 News

Clarion news for March 2001.

Posted Thursday, March 01, 2001

Dynamic Filters: The Theory Behind the Facts

by Steven Parker

Published 2001-03-27

I think that programmers often make things more complex than they need to be. This was forcefully demonstrated to me some years ago when I asked several Clarion programmers how they would find the number of characters in a string (i.e., the length of that part of the string actually containing data).

Most solutions ran a dozen or so lines of code, well formatted. Typically, the submitted code looped from the end of the string to the first non-blank. Most used the `Sub` function but a few did take advantage of Clarion's string slicing ability (which is not only much more efficient than `Sub` but also makes the code much easier to read). One solution was over one-half page of code, not so well formatted. Well formatted, it ran two thirds of a page.

`Len(Clip(myVar))` was all that was required to achieve the objective.

It wasn't that these developers, at least not all of them, were unaware of the `Len` or `Clip` statements. It was, I believe, that programmers are trained to look for complex solutions to problems. Instead of using simple built-in functions, these developers adopted and adapted a variety of algorithms.

While I would not expect current Clarion developers to create so much extraneous code in finding the size of a string, setting up filters has provided new opportunities for introducing complexity, complexity that is entirely unnecessary.

For example, I recently saw an attempt to filter by creating a queue to store variable labels and the values the end user selected for each. The queue was intended to be passed to another procedure and, there, decoded to determine whether or not a specific record was to be included in a browse.

When appropriately used, queues are an extraordinarily powerful tool (in fact, there was a newsgroup posting, not long ago, in which a Clarion developer claimed that queues are the reason he refuses to drop Clarion - hyperbole, perhaps, but this sentiment is indeed

shared by many of us). But to effect a filter in the manner described is clearly neither an appropriate use of queues nor a very good use of resources. Decoding the queue would require extensive use of `Who` and/or `What` just to extract the field label from the queue *and* the file structure before the values could be compared. Just trying to sketch the strategy required by this scenario gives me a migraine.

Another recent case involved setting up a large single-dimensioned array; the filter code compared a file variable to each array element. Well, I do use arrays occasionally but I have a long-time mental block when it comes to them (this is probably due to the fact that my first computer course was in FORTRAN and what we now call "arrays" were handled by nested DO loops and I never managed to DO it the correct order).

Implementing a dynamic filter simply shouldn't be this complicated.

The great secret(s) of filtering

There are three and only three things you need to know to effectively implement a runtime filter:

(1) The user interface is irrelevant

(2) There are only two ways to code a filter in ABC

and, the greatest secret of all:

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

It *is* just that simple.

The interface is irrelevant

Too many developers get hung up worrying about the user interface that will be created for capturing data from the user for use in the filter. In fact, not counting the QBE template introduced in C5, there are only three possible user interfaces.

First, there are multi-tabbed browses, each of which may have a unique filter.

Second, there are simple data entry controls, like entry fields, file drops and radio buttons. These may be populated on a tab, or a process or report's Progress window (for entry and display). For more on this technique see Conditional Sort Orders and Page Breaks in Reports Part 1. Figures 1 and 2 provide a summary of the technique.

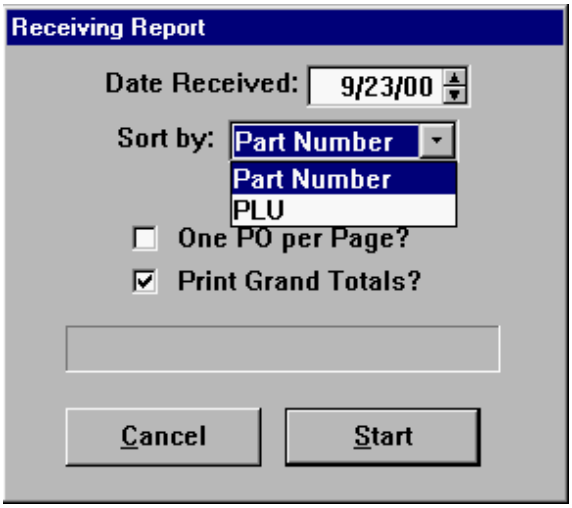


Figure 1. Using the Progress window to capture filter variables

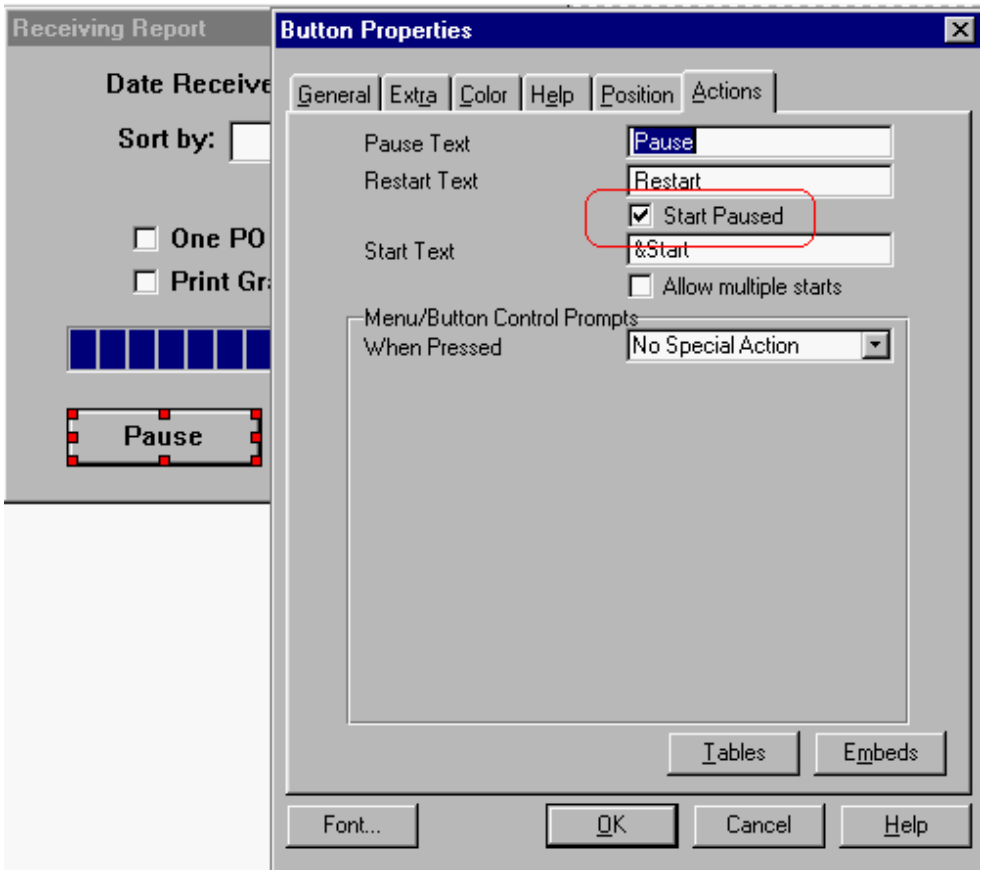


Figure 2: Setting up the Pause the Process Control Template

Third, there is what I call a "criterion window." This is a separate window procedure with one or more (usually "more") data entry controls.

That's it. And, since the first allows the user no control over the filter and, therefore, over what is displayed, it doesn't really count. In fact, what I have just described is all pretty standard window design stuff. No rocket science here. Any Clarion developer who has completed the tutorials can create any of these user interfaces.

So, the user interface is just *not* a significant obstacle to be overcome.

How you declare the variables used in the filter can be quite important (do you declare them globally, in the criterion window and pass them to the target procedure or in the target procedure and pass them to the criterion window, etc.) but the interface itself is easy and just not relevant to the implementation of the filter itself.

There are only two ways to filter in ABC

The ABC templates homogenize procedures and their embeds. That is, the same embeds are available in virtually all ABC procedures and this is most obviously true in procedures where filters are used (browses, processes and reports). The Legacy templates do not sport this feature. For example, Legacy processes have an "Activity for Each Record" embed but reports and browses do not. Legacy browses have an "After Range and Filter Checking" embed that neither processes or reports have.

ABC browses, processes and reports all have `INIT`, `TakeRecord` and `ValidateRecord` methods (`TakeRecord` is not recommended for filters but is an option). In any of these embeds, I can effect a filter. All have a Record Filter prompt on the Procedure Properties screen. Browses have one for each tab in which a filter may be created (see Procedure Properties | Browse Box Behavior | Conditional Behavior | Properties).

(ABC browses, processes and reports are structured with remarkable similarities.)This uniformity of embeds certainly justifies the assertion that browses, processes and reports can *share* filtering techniques. On its face, however, it appears that there are multiple ways of creating a filter, using the embeds mentioned.Appearances notwithstanding, there are, in fact, only two ways to create a filter in the IDE:

> (1) Tell the underlying View how to filter
>
> (2) Write embed code to filter on a record-by-record basis

That's it.

Affecting the view

When you complete the Record Filter prompt, as shown in Figure 3, you cause a SetFilter statement to be generated, as in Figure 4.

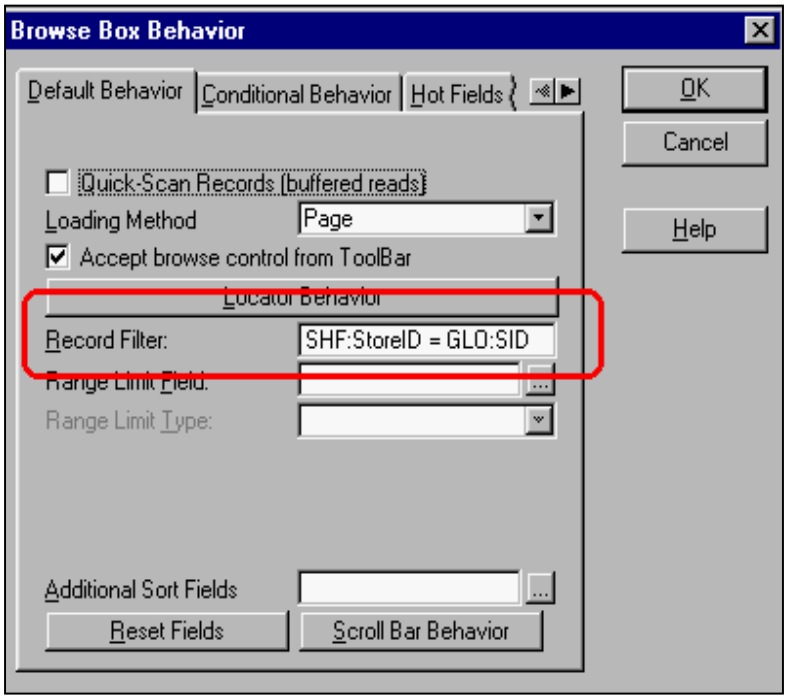


Figure 3. Standard record filter

```
OPEN(QuickWindow)
SELF.Opened=True
! [Priority 8030]

! Filling in browse options
BRW1.Q &= Queue:Browse:1
BRW1::Sort1:StepClass.Init(+ScrollSort:AllowAlpha+S
BRW1.AddSortOrder(BRW1::Sort1:StepClass,SHF:PartNum
BRW1.AddLocator(BRW1::Sort1:Locator)
BRW1::Sort1:Locator.Init(?SHF:PartNumber,SHF:PartNu
BRW1.SetFilter('(SHF:StoreID = GLO:SID)')
BRW1::Sort2:StepClass.Init(+ScrollSort:AllowAlpha+S
BRW1.AddSortOrder(BRW1::Sort2:StepClass,SHF:FormatK
BRW1.AddLocator(BRW1::Sort2:Locator)
BRW1::Sort2:Locator.Init(?SHF:Format,SHF:Format,1,B
BRW1.SetFilter('(SHF:StoreID = GLO:SID)')
BRW1::Sort0:StepClass.Init(+ScrollSort:AllowAlpha+S
BRW1.AddSortOrder(BRW1::Sort0:StepClass,SHF:PLUKey)
BRW1.AddLocator(BRW1::Sort0:Locator)
BRW1::Sort0:Locator.Init(?SHF:PLU,SHF:PLU,1,BRW1)
BRW1.SetFilter('(SHF:StoreID = GLO:SID)')
```

Figure 4. Code generated by the Record Filter prompt

The ABC Reference explains that the `SetFilter` method impacts the view:

> The **SetFilter** method specifies a filter for the active sort order. When the filter is applied, *the view only includes* those elements whose expression evaluates to true. (Italics added.)

The `SetFilter` method takes "A string constant, variable, EQUATE, or expression that contains a FILTER expression." Developers most commonly use a constant, entering the desired filter in the Record Filter prompt, as show in Figure 3. Variables, prepended by "!" are

also common, and to use SetFilter this way I need only remember to prime the variable before using it (otherwise I get no filtering, though if the procedure is a browse I can prime the variable later and reset the browse).

Because the SetFilter method is neither protected nor private, I can call it directly in an embed. In an earlier version of my knowledge base, for example, I used buttons rather than tabs to let the user filter the results. Each button primed a variable (FilterString) and I called the following routine:

```
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'')')
   Of 'DCT'
     BRW1.SetFilter('(Clip(Upper(FAQ:Product)) |
       = ''DCT'')')
   Of 'SQL'
     BRW1.SetFilter('(Clip(Upper(FAQ:Product)) |
       = ''SQL/ODBC'')')
   Of 'C5/EE'
     BRW1.SetFilter('(Clip(Upper(FAQ:Product)) |
       = ''C5/EE'')')
   End
 End
 ThisWindow.Reset(1)
```

(I based the syntax on what the code generator did with the Record Filter prompt. Otherwise, I would never have figured out the correct sequence and quantity of apostrophes.)

Manually calling SetFilter has an additional, powerful use. If several tabs on a browse need to share the same filter, you may not want to complete the Record Filter prompt for each tab. For example, Figure 5 shows a browse with a shared filter.

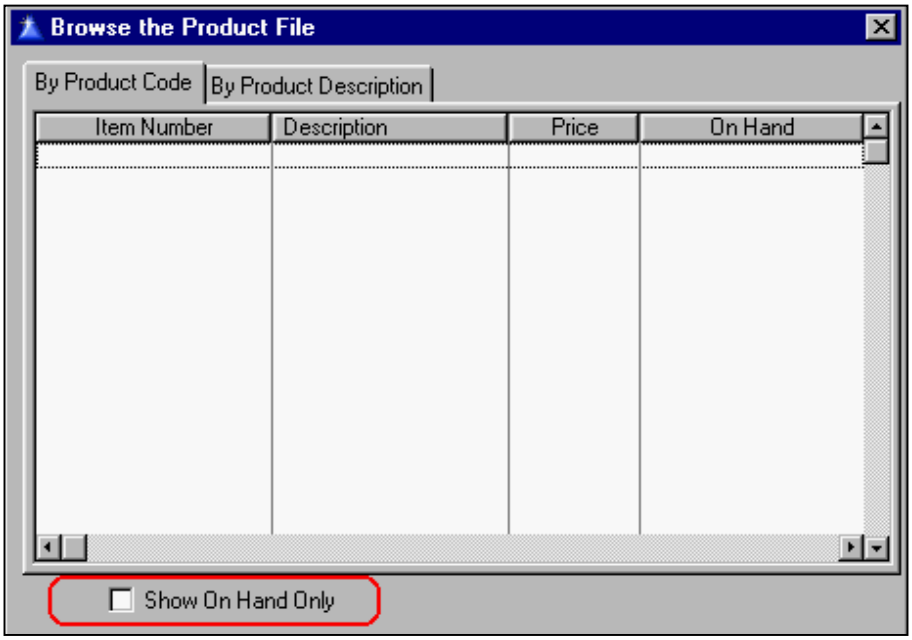


Figure 5. Browse with shared filter

In this case, calling `SetFilter` directly can permit you to filter multiple tabs without repeatedly entering the filter in the tab's conditional properties:

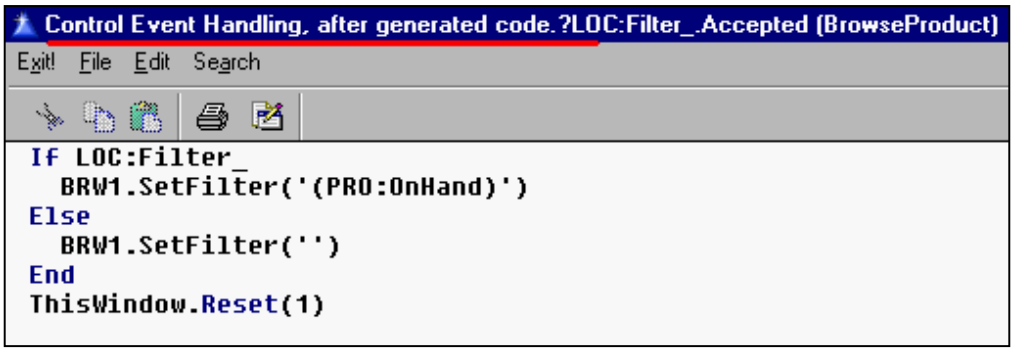


Figure 6. Code driving a shared filter

In this case, because the filter can be set from a checkbox, I had to use the checkbox's Accepted embed to complete the `SetFilter` statement. Then, in the sheet's `NewSelection` embed, I ensure that the control's value is read on each tab change:

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

In other cases, code in the `NewSelection` embed is sufficient to ensure that several tabs share a single filter.

Record-by-record filtering

Record-by-Record filtering, in its simplest form, looks like this:

```
Loop Until Access:myFile.Next()
  If <condition>
    Return Record:Filtered
```

```
  Else
    Return Record:Ok
  End
End
```

You usually implement this kind of filtering in the `ValidateRecord, Before Parent call` embed, as shown in Figure 7.

Figure 7. Typical ValidateRecord code

Note that functions called from this embed do not have to be `Binded`. This allows developers to construct extraordinarily complicated filters.

If you decide to execute your code after the parent call, set `ReturnValue` instead of doing a `Return Record:Filtered` or `Return Record:Ok`, as the default code makes clear:

```
ReturnValue = PARENT.ValidateRecord()
  ! [Priority 5050] embed
  BRW1::RecordStatus=ReturnValue
  IF BRW1::RecordStatus NOT=Record:OK |
    THEN RETURN BRW1::RecordStatus.
  ! [Priority 6600] embed
  ReturnValue=BRW1::RecordStatus
  ! [Priority 9000] embed
  ! End of "Browser Method Code Section"
  RETURN ReturnValue
```

(If you are using the Legacy templates, you will set `ReturnValue` and then `Exit`, as `ValidateRecord` is a Routine.)

This method is typically used for conditional filters; that is, filters that depend on a condition being met first:

```
!if there is a followup date
If LEA:FollowupDate
  ! and it is within 30 days
  If LEA:FollowupDate => Today() + 30
    Return Record:OK
  End
End
```

Another typical use is when something must be written to the file when a record qualifies, particularly within a Report:

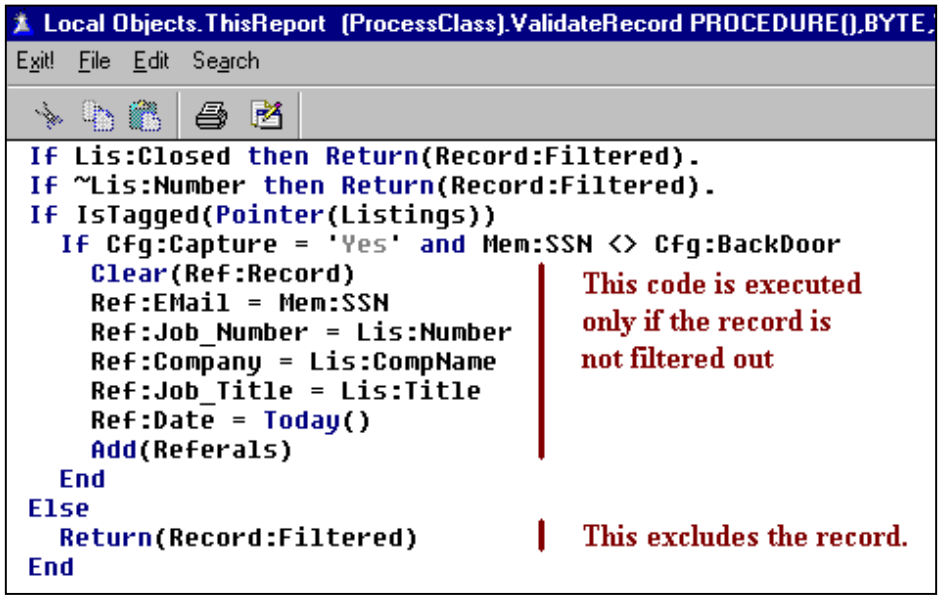


Figure 8. ValidateRecord with file update

Now, it may seem that checking each record is the method of choice when there are very complex filter conditions (too complex to be comfortably fit into a string), when filtering is done in a function, when files must be updated or when a single record must meet multiple conditions.

There is, however, another set of conditions under which this method is preferable over the `SetFilter` method.

The `SetFilter` method works by filtering the view, similar to declaring the view with a filter:

```
ViewOrder VIEW(Customer),Filter('CUS:Balance > 0')
```

So, before your procedure can process records, the runtime library has to construct the view.

While the view is built, the progress window in a report or process displays nothing. A browse is empty during this time. As view's data grows and/or the relative number of qualifying records decreases, the progress window's failure to display anything may cause users to think that the app has hung.

Similarly, if there are few qualifying records, even in smaller files, the progress window may "hang" for a few seconds then either jump directly to 50% and stay there for a while, sometimes quite a while, and then zip to 100%. The progress bar may also go from zero to 100 in nothing flat and "hang" at 100% for several seconds.

In either case, the end user does not see things as they expect to see them. It may well be necessary to trade off performance for perception (and, yes, there are rare cases where a full loop is faster).

Filters can include or exclude records

When you're reading all records, the greatest secret of filtering. (And the construction of a view must, at least at first, go through all records – unless the filter can use a key – for flat files; with SQL databases, the view is constructed by the server.)

As a record is read, do I want to include it or do I want to exclude it? Of course, the answer is "Yes." I can return one of three values after reading a record:

```
Record:Ok
Record:Filtered
Record:OutOfRange
```

`Record:Ok` includes a record. `Record:Filtered` excludes a record. `Record:OutOfRange` excludes a record and terminates processing.Understanding what you are trying to accomplish can dictate which of these you want to return.

For example, if a record must meet all of a number of conditions, the first condition not met is determinative (a conjunction can be fully resolved at the first false value). If a record must meet one of several conditions, the first condition met is determinative (a disjunction can be resolved at the first true value).

Let's say that I want to create a tickler (reminder) browser on a real estate property file. I want to see any record where I have a follow-up date, the lease is expiring in the next 90 days or there is a notice date within the next 30 days. The key here is the word "or:"

```
If PRO:ContactDate <= Today()
   Return Record:OK
End
If LEA:ExpDate - Today() <= 90
  Return Record:OK
End
If LEA:NoticeDate - Today() <= 30
  Return Record:OK
End
! Record met no criteria
Return Record:Filtered
```

Notice that I can also check fields in a related file (the Relation Manager handles retrieving them for me when the view is built). But, the important part is that a record meeting *any one* of three conditions is to be included. Therefore, I can make a determination when the first one is met. Only if all three checks fail do I exclude the record.

On the other hand, if I collect a series of values from a criterion window and a record must meet all of them, the first non-match is fully determinative:

```
If UNI:LCS <> LOC:LCS
```

```
   Return Record:Filtered
End
If UNI:DLS <> LOC:DLS
   Return Record:Filtered
End
! All conditions met
Return Record:Ok
```

By carefully analyzing the fastest way to make a logical determination, you can minimize the performance penalty of looping through the whole file. Well constructed, this kind of filter can be very fast.

Summary

Clarion's templates hide complexity. The ABC classes are supposed to also. But too often we developers re-introduce complexity.Once you realize that filters can be created in only two ways, it is really much easier to optimize filter expressions and performance.

Next time, I'll look at actual implementations of filters.

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

Search
Go

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

The Clarion Advisor: Another Use of Loops

by Dave Harms

Published 2001-03-26

This edition of the Clarion Advisor is about the Clarion LOOP construct. Most developers use LOOP to iterate through the contents of a queue or table, or to perform some repetitive calculation. But LOOP can also simplify IF statement processing. I don't know about you, but every once in a while I come across a block of code that looks like this:

```
IF ConditionA=True AND ConditionB=False
  IF IsThisDataOkay()=True
    IF CheckForUnexpectedError()=True
      ! Whoa! Get out! Now!
    ELSE
      ! do something else
  ELSE
    IF IsThisDataBad()=True
      ! Need to get out of here too!
  END
  ! some more code
END
```

The problem with the above code is that under some circumstances I want to immediately exit from somewhere inside that twisted logic. I guess I could use a GOTO to jump to the end of the code, but I really don't want a reputation as a GOTO coder. What I'd really like is a BREAK statement in my IF logic, but since there isn't one, I'll adapt a LOOP.

To give myself a quick exit out of any block of code, I wrap that code in a single-pass loop:

```
LOOP 1 TIMES
  ! some code
  IF DisasterStrikes() THEN BREAK.
  ! some more code that will otherwise execute
END
```

There are several other ways to code this loop. I can put a break statement just before the loop end:

```
LOOP
  ! some code
  IF DisasterStrikes() THEN BREAK.
  ! some more code that will otherwise execute
  BREAK
END
```

or I can use a WHILE statement that evaluates to false, terminating the loop after one iteration:

```
LOOP
  ! some code
  IF DisasterStrikes() THEN BREAK.
  ! some more code that will otherwise execute
WHILE FALSE
```

The first approach is the most readable, since anyone coming across this code immediately understands that this loop is only supposed to have one iteration. And although I don't need this technique often, I've found the real benefit is readability. I can often rewrite my logic into a more linear set of single IF blocks rather than nested IFs. That makes my code a lot easier to maintain.

Procedure prototypes redux

Following last month's discussion of procedure prototyping, Arnor Baldvinsson offered a few additional tips. When a procedure returns data, Arnor copies the value from the prototype prompt, which may look something like this:

```
(STRING pString),LONG
```

When he pastes this value into the parameters prompt, Arnor comments out the return data type:

```
(STRING pString) !!,LONG
```

This means that both the map and the procedure declaration will be (almost) identical, and both will show the return type. Arnor also copies the prototype as a comment into a convenient procedure embed:

```
!! (STRING pString),LONG
```

That way the prototype shows up in the embed tree. Thanks, Arnor!

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

Clarion MAGAZINE

Search
Go

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

Book Review: The Art & Science of Web Design

by Dave Harms

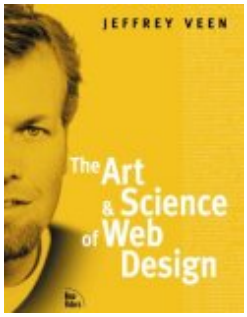
Published 2001-03-20

Reviewed by: Dave Harms

The Art & Science of Web Design

by Jeffrey Veen

New Riders, 2000; 260 pp; ISBN: 0789723700



When Clarion made the move from DOS to Windows, Clarion developers had to learn a lot of new concepts to effectively deploy their applications on the Windows GUI. I don't think it's an understatement to say that the move to Web development presents challenges just as significant as the move from DOS to Windows.

There are many different flavors of Web development. Perhaps you've created a Web site with a few HTML pages, solely to promote your consulting services. It could be you're deploying large-scale Web applications, using Clarion or other technology. Or perhaps you're somewhere in the middle, adding interactivity to a mostly static site. Chances are you're already a stranger in Webland, or you will be soon.

I recently picked up Jeffrey Veen's book *The Art & Science of Web Design*. While the material is particularly appropriate to e-publishing ventures like Clarion Magazine, it contains a lot of good information that applies to almost any kind of Web development. Veen's credentials are impressive; he cut his teeth developing Web sites such as HotWired and HotBot, is involved with the World Wide Web Standards Consortium, and writes for WebMonkey. He is also the author of the book *HotWired Style: Principles for Building Smart Web Sites*.

Veen begins his discussion of Web design with a look at the history of the Web. If you've read any Web-related books at all you've probably encountered all to many such historical summaries, but this one is worth reading. Veen is more concerned with the evolution of typesetting and document retrieval than with the development of browser technology. The Web, he says, is a collaboration of words, pictures, and code, and he elaborates these into the concepts of structure, behavior, and presentation. Text is particularly important because, although visually limiting, it is universal, fast, and above all, machine readable. You can convey a great deal of information through images, but it's almost impossible to categorize, search for, and retrieve non-textual data.

Chapter Two covers interface consistency. There are certain fundamentals about navigating the Web which you tamper with at your peril. For instance, Web surfers understand that underlined text is a hyperlink. If you underline non-linked text, or remove underlines from your links, you run the risk of confusing or alienating your users. As Veen explains, Web interfaces are often unavoidably confusing. Say a user visits your Web site and clicks on a link to one of your pages, drilling deeper into your Web site. On that page you've added a "back to the main page" link. The user clicks this link, and is back up at the "top" level. Now what happens if the user pressed the "Back" button? It's back to the "lower level" page. Conceptually the user has used "Back," which has the general meaning of going back to a higher level, to go to a lower level. And the user didn't have the choice of going forward because the "Forward" button was disabled. So how do you deal with this problem?

The answer is to give the user a context by exploiting page layout. The classic approach is the three-panel design (of which Clarion Magazine pages are one example). These panels answer the questions: "Where am I?" "What's here" and "Where can I go?" The top panel is the brand area that tells you where you are. The left panel is the navigation area, and the right panel is the content area. There are of course many variations on this theme, and many subtleties of design to consider.

Next, Veen delves into the architecture of Web site information. This chapter is more appropriate to content-rich sites than to, say, small Web applications. How do you organize the material on a site so that it's readily available to users? Veen looks in some detail at the difficulties faced by the large portal sites as they index and present a vast array of Web resources.

Chapter Four discusses the behavior of Web sites. Veen points out that while a print publisher knows at ship time the location of every pixel in the product, Web developers don't have that luxury. When you develop a Web site, you deliver code, and how that site appears can vary widely from user to user. There are some approaches you can take to minimizing the problem, and these fall into client-side (such as JavaScript) and server-side approaches. On the whole, Veen favors server-side solutions to browser incompatibilities, and in Chapter Five offers an understanding of the scope of browser issues and some strategies for accommodating the large number of different browsers (more than 100) currently in use.

In Chapter Six, Veen discusses speed, and the illusion of speed. Small images are a byword in the industry, but even when pages are large there are techniques you can use to make a page appear to load more quickly. Judicious use of HTML tables can make a difference, as can exploiting absolute positioning in cascading style sheets.

Chapter Seven is more or less a rant against the current state of internet advertising. Veen isn't opposed to advertising, but he does have a lot to say about how it could be done, and isn't.

Veen closes the book with a discussion of object-oriented publishing, including an example of a small ASP database application. To title this chapter "Object-Oriented Publishing" is a bit of a stretch, since there isn't anything overtly OO about the overall design; a procedural scripting language would work as well. More seriously, there's no discussion of the importance of separating presentation logic from application logic, and the ASP page is rife with embedded code. Still, Veen is attempting to do in one chapter what could easily fill a large book, and compromises are inevitable.

Throughout this book you'll find snippets of code, but this is definitely not a coding guide. It is, as the title suggests, a thoughtful and thought-provoking discussion of Web design concepts and practices.

The real question, of course, is how appropriate this book is to those of us doing Clarion Web development. If you're primarily trying to deliver desktop applications across the Web, using ClarioNet or Citrix, then you'll find the book an engaging read, but that's about all. On the other hand, if you're at all interested in SoftVelocity's ASP templates and other server-side Web technologies, or even if you just create your own HTML pages, this book should be on your reading list.

For more information visit the book's home page:
http://www.veen.com/artsci/

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

Clarion MAGAZINE

Clarion News

SysList 1.1 Released

SysList 1.1 is now available from solid.software. New features include drag and drop and custom font support, and improved mouse event handling. This update is free for registered users. A new demo version is also available

Posted Wednesday, March 28, 2001

VariView Adds Message Streaming

The VariView templates now have a tab on the toolbox for messages. One of the original uses of VariView was to replace the need to add Message() to code to see the values of fields and variables, but this doesn't work well in a loop. The VariView templates now let you use a function to add variable values to a list box so you can see the changes in data as the loop progresses. Other features/enhancements include optimized tab refresh and iconizable toolbox.

Posted Wednesday, March 28, 2001

Open Clarion News Server

Ron Schofield has set up a news server with a few newsgroups for discussing the Open Clarion project and other RAD topics.

Posted Wednesday, March 28, 2001

PickDBG Debugger Article

This month's article from Paul Attryde covers the PickDBG

program, installs as the system debugger and allows you to determine at GPF time which debugger you would like to run to debug the GPF. Particularily useful for those of you running multiple copies of Clarion or doing application development in different development tools.
Posted Wednesday, March 28, 2001

Free Auto Build Number Template

Benjamin Dell has created a freeware template to set an auto build number for an application. This extension template requires changes to the ABC template. Use at your own risk.
Posted Wednesday, March 28, 2001

Clarion 5.5c Released

Clarion 5.5c is available for download. This update fixes the regression in 5.5b for Message box text wrapping, and adds many new fixes and enhancements. Less than 1 MB. Requires C5.5b.
Posted Monday, March 26, 2001

Keystone Reduces Pricing

Effective immediately, Keystone Computer Resources has reduced the price of all of its Clarion Tools, Templates, and Utilities. Purchase NetTools, Screen Capture Tools, Queue Edit-In-Place and other products for just $99.00 USD each.
Posted Friday, March 23, 2001

Imaging Templates 1.10 Released

Nextage Consulting has released version 1.10 of the Imaging Templates. New features include: installer supports C55b; corrected issue with pasting an image from the clipboard into a blank image; changed Create Directory to use an internal function instead of shelling to DOS; added support for client/server data sources in the CurrentImagePath procedure; added export to AWD/BMP support for legacy templates. Existing customers will received an email notice with download instructions.
Posted Friday, March 23, 2001

Web Site Of The Week

Okay, we don't really run a Web Site Of The Week feature at Clarion Magazine, at least not yet. But if we did, Joel on Software would be one of the first. What's it all about? As Joel says, "Over the years I've formed some pretty strong opinions about software development. Friends and colleagues who have heard my rants have often encouraged me to write them down."There is a wealth of information here for developers and development managers of all stripes. Many thanks to Carl Barnes for pointing this gem out. For a list of all articles on this site see http://joel.editthispage.com/stories/.

Posted Thursday, March 22, 2001

Buggy 2.1.1 Available

An update to the Buggy bug tracking tool is available to all registered users. The trial version is also being updated.

Posted Wednesday, March 21, 2001

SoftVelocity FAQ System

SoftVelocity has an initial release of its online FAQ system up and running. Comments and contributions are welcome.

Posted Tuesday, March 20, 2001

Search Engine Profile Exchange Updated

Encourager Software has created an centralized information resource for software authors that need to submit their programs and web sites to the various search engines.

Posted Tuesday, March 20, 2001

VariView 2.00 Released

VariView 2.00 is now available. Features include: save to file; contents of variables and files increased to 2048; option to view the full value of a field; fixes for details of removable drives in Environment tab. A new demo is available.

Posted Tuesday, March 20, 2001

Linder SetupBuilder 3.51 Available

Version 3.51 of the popular Linder SetupBuilder installation tool has been released. This release contains several bug fixes and feature enhancements, including: verification of downloaded data files using the MD5 hashing algorithm; ability to delete locked files; silent installs; support for file groups; numerous new variables (32 bit); new file installation options. SetupBuilder offers a user-friendly visual development environment that does not require knowledge of a script language. The update from Linder SetupBuilder 3.50 to 3.51 is free to registered users. Regular cost is $169.00 for a royalty-free usage license. A fully functional trial version is available.

Posted Tuesday, March 20, 2001

Free G-Notes For Clarion Editor

Gitano has released a free version of G-Notes for the Clarion Editor. G-Notes lets you keep all your code snippets and application notes handy. This release has been compiled with the latest (C55 b) and supports CW2003, CW4, CW5PE, CW5EE, C55PE and C55EE.

Posted Monday, March 19, 2001

Clarion MAGAZINE

Web Splash Screens

by Carl Barnes

Published 2001-03-20

It's common for an application to need to open a browser window to display a web page. For example, from the Help menu of your application you might want to link to your support web page. The easy way to do that is by calling `ShellExecute()` with the URL, and let the Windows Shell figure out what to do with it. In this article I will look at some ways of using ShellExecute with what I call a "Web Splash Screen" to provide a better user experience.

`ShellExecute` is easy to use and so I won't discuss it in detail. The included sample application will give you the prototypes and example code for using `ShellExecute` to open a web page from a Clarion application. To make things easier I've created a wrapper procedure for `ShellExecute` called `OpenURL`:

```
OpenURL     PROCEDURE(STRING Url)
ShellFile         CSTRING(256)
            CODE
    ShellFile = CLIP(Url)
    ShellExecute(GetDesktopWindow(),,ShellFile,,,5)
```

Once you define this procedure, from within your code you simply call it like this:

```
OpenURL('www.mysite.com')
```

The Internet, like Heinz ketchup, can be S-L-O-W

One of the problems with the Internet is that opening a new URL can be slow. What I mean is when you call `OpenURL('www.mysite.com')` Windows will open a browser window quickly, but it might take many seconds for the browser to actually connect to your website and show the web page. During this waiting period there is no telling what the user might be viewing in the browser window. It could be a blank page, or the user's home page, or the previous page. This can be confusing to some users because there's no obvious indication that the browser is trying to open a page on your site. What I wanted was some control over what was displayed during this waiting period.

Web splash screens

A web splash screen is simply an HTML file you install on the users hard drive that redirects the user to your website. Then instead of using `OpenURL('www.mysite.com')` you open your splash screen file with `OpenURL('mysitesplash.htm')`. The advantage is that this page opens instantly and the user gets immediate feedback that he is going to your site. You can create your splash screen with any HTML tool and make it as nice looking as you want. Below is an example of a simple web splash screen that asks the user to "Please wait while we connect..."

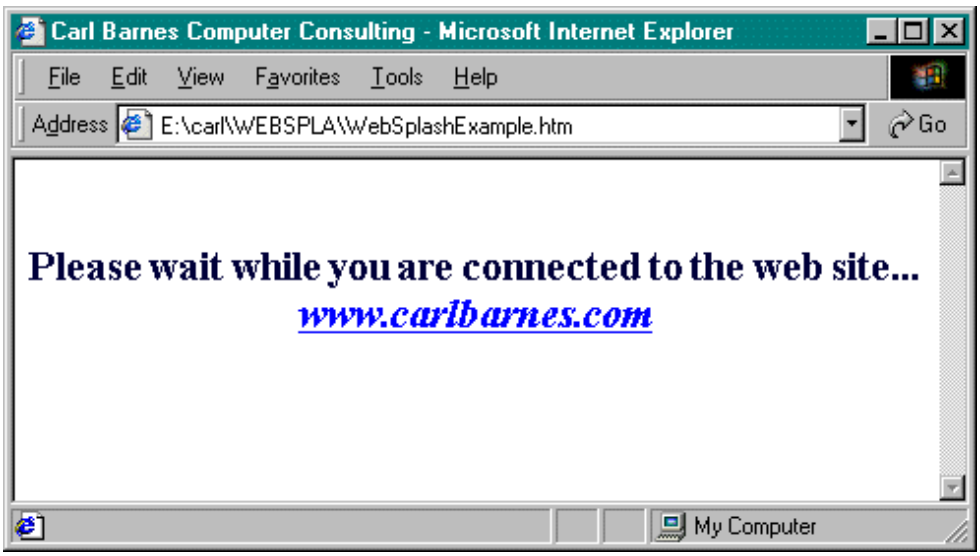


Figure 1. A Web splash screen

The HTML for this splash screen is fairly simple:

```
<html><head>
<title>Carl Barnes Computer Consulting</title>
<meta http-equiv="REFRESH" content="1;
  URL=http://www.carlbarnes.com/">
</head>
<body bgcolor="#FFFFFF" text="#000040">
<center><br><H2>
Please wait while you are connected
to the web site...
<i><A href=http://www.carlbarnes.com/>
  www.carlbarnes.com</A></i></h2>
</center>
</body></html>
```

The magic that makes this work is the second line with the `meta http-equiv="REFRESH"` command. This tells the browser to wait 1 second, then refresh the web page and load the URL www.carlbarnes.com. For testing you might want the wait time set to 15 or more seconds so you can see what's happening, but for release you should set it to zero so the user is not delayed. It is usually suggested that all pages that perform this type of redirection also provide a link for the user to click on in case the

browser does not support the refresh tag or redirection has been optionally disabled. The second to last line has an "A HREF" tag with the hyperlink the user can click on.

What if the link is a variable?

Recently I ran into the need to append variable parameters to the end of the URL. I wanted my application to load the support web page and pass the user's serial number and version, e.g. `OpenURL('www.mysite.com/support.asp?sn=12345&ver=3.20')`. If I wanted to use a Web Splash, I would have to write out a customized HTML file from my Clarion program. I had two options:

1. Embed the HTML code in my program
2. Read a "template" HTML file and write the customized output file

If my splash screen was going to be small and simple, I would choose option 1. But if I was going to go to some significant HTML design work, I would choose option 2 and keep all of the HTML code external to my Clarion code. My client had used FrontPage to make a professional looking page with nice graphics, and so option 2 was my only choice.

Using PUTINI with HTML files

I thought about loading an ASCII file, doing some `INSTRING`'s, some slices and writing out the resulting ASCII file, but that was more work than I wanted at 5 p.m. when this requirement came up. I kept looking at the HTML code I had to change `("URL=http://www..."}` and thinking how much that looked like an INI file. I wished I could just do a `PUTINI('','URL'...)`.

INI files are pretty simple: you must have a `[Section]` and an `Entry=Value` that starts at the beginning of a line. HTML files are pretty flexible; they don't care about line breaks and allow hidden comments. By editing the HTML file, inserting INI sections within comments, and adding line breaks where required, I came up with the following. This file displays in a browser, and I can still update it using `PUTINI`.

```
<html><head>
<title>Carl Barnes Computer Consulting</title>
<!-- need [URL] for PUTINI
[URL]
BaseURL=http://www.carlbarnes.com/
-->
<meta http-equiv="REFRESH" content="1;
URL=http://www.carlbarnes.com/
"></head>
<body bgcolor="#FFFFFF" text="#000040">
<center><br><H2>Please wait while you
are connected to the web site...
<i>
```

```
<!-- need [HREF] for PUTINI
[HREF]
-->
<A
href=http://www.carlbarnes.com/
>www.carlbarnes.com</A>
</i></h2></center></body></html>
```

HTML comments begin with <!-- and end with -->. Anything you put between these comment delimiters won't be used for display, so the browser ignores the INI file section header. Browsers also ignore line breaks (except inside <PRE> formatted sections) so it's easy to embed the INI data inside the HTML.

To customize the HTML file with the customer serial number and version I use the following Clarion code. PUTINI creates text in the form entry=value. This works because some HTML tag bodies use an identical format.

```
CustomURL=|
  CLIP(GETINI('URL','BaseURL','','.\SplashCB.HTM')) |
  & '?SN=' & CLIP(SerialNo) &'&Ver=' & CLIP(Version)
PUTINI('URL','URL', CustomURL,'.\SplashCB.HTM')
PUTINI('HREF','href', CustomURL,'.\SplashCB.HTM')
OpenURL('SplashCB.HTM')
```

I also hid the URL that all of these links will be using in the HTML comment. That way I can use GETINI to retrieve that value from the HTML file. That's one more piece of HTML data I can keep out of my Clarion code.

Splash open any URL

The HTML splash screen concept can easily be extended to open any URL with a generalized splash screen HTML file. Besides the advantages discussed above, the Splash Open method insures that a browser window will be opened by ShellExecute. If your application will allow your users to specify URLs, you risk a typo that might cause ShellExecute to open an unexpected application.

Be aware that while you can call ShellExecute('www.abcd.com') and Explorer will add the http:// protocol identifier (because the URL conforms to a pattern common in HTTP requests), within an HTML HREF you must specify the protocol or the HREF will not work. So if you are going to PUTINI a user entered URL into an HREF or URL you should verify that the URL contains a protocol. Rather than code for every possible protocol I just check for the colon double backslash e.g. INSTRING(' ://',Cus:HomePage,1).

Summary

Web splash screens ensure that your user will never see an unexpected page when linking to the web from your application.

You also can give your application a more professional web interface. To get you started, I've created a sample application that demonstrates the concepts above and includes a generalized Splash Open procedure and HTML file.

Download the source

For more information

ShellExecute:
http://msdn.microsoft.com/library/psdk/shellcc/shell/Functions/ShellExecute.htm

HTML Redirection:
http://internet-tips.net/HTML/META_httpequiv_refresh.htm

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

Reader Comments

Add a comment

The Clarion Challenge Results: Useless Tab Text

by Dave Harms

Published 2001-03-14

Several weeks ago I tried to plumb the meaning of inverted text on sheet tabs, and in the process came up with a small but useless application that sent tabs scurrying around all four sides of a sheet. I posted my code and challenged ClarionMag readers to fix a glaring bug in the code, write the code more compactly, and finally to come up with a real application for inverted tab text.

On the third point, **Carl Barnes** suggested that a Rolodex wheel might possibly need two sheets, one with inverted text tabs, to give the image of a flipped card. "Another use would be to invert text to denote tabs that require programmer attention before final release. e.g. when working on a Wizard it is handy to have the tabs showing and some times it gets released that way. Turning the text upside down might draw attention to it from the test department. Also it's easy to search for "UP,DOWN" if you have a good search plan and program." A good search program. Hmm. I wonder where I could get one of those...

Carl also successfully solved the problem of the inverted text that wouldn't go right side by guessing that `PROP:Angle` would do the trick. As a result, Carl only needed to change:

```
?sheet{prop:upsidedown}=0
```

to:

```
?sheet{prop:angle}=0
```

After the fact, Carl came up with the following code which takes a more methodical approach and examines all of the available sheet properties, then tests to see what changed after the text was inverted:

```
CheckProps   routine
    DATA
!Property save queue
pq      queue,pre(pq)
prop        long
```

```
value       string(32)
        end
pidx            long
comparevalue    like(pq.value)

    CODE
    free(pq)
    !save all the properties in a queue
    loop pidx = 07a00h to 07effh
         clear(pq)
         pq.prop  = pidx
         pq.value = ?sheet{ pidx }
         IF  pq.value  THEN ADD( pq ).
    end

    !change the property that is a mystery
    ?sheet{prop:upsidedown} = 1

    !now see what changed
    loop pidx = 07a00h to 07cffh
         clear(pq)
         pq.prop  = pidx
         GET(pq, pq.prop)
         comparevalue = ?sheet{ pidx }
         IF pq.value <> comparevalue
            message(pidx & '|old=' & clip(pq.value) |
              & '|new=' & CLIP(comparevalue) )
         end
    end

    ! pidx=31977 (7CE9h) changed from
    ! 0 to 1800, which is PROP:Angle
```

As Carl points out, "a lot of people forget that PROP:xxx are just LONGs and so property changes are just LONG{LONG}=Value."

Carl's code for moving the tabs around is also considerably more compact than mine, and works with any number of tabs, not just four.

```
IF ~INRANGE(Tab+TabIncr, 1, NumTabs )
   !move to next side
   side = side % 4 + 1
   !Adjust tabs for specific side
   EXECUTE side
       TabIncr = 1
       Tab = 1
       TabIncr = -1
       Tab = NumTabs
   END
   ?sheet{ CHOOSE(Side,prop:above,prop:right,|
      prop:below,prop:left) }=1
   ?sheet{prop:angle} = CHOOSE(Side,0,2700,1800,900)
```

Brice Shagane didn't solve the inverted text problem, but weighed in with some nifty bitwise code to shift the tabs around.

```
if count <= 9
   tab = count - BAND(count-1, 100b)
else
   tab = BXOR(count, 11111b) - 14 - |
     BAND(BXOR(count, 11111b)-15, 100b)
end
execute (tab)
   select(?tab1)
   select(?tab2)
   select(?tab3)
   select(?tab4)
end
count += 1 - BAND(count, 10000b)
```

Awesome. I'm still trying to uncross my eyes.

And finally, **Gordon Smith** dropped off a very compact version of the program, shown here minus the window structure:

```
count byte(1)

   code
   open(window)
   accept
     if event() = event:timer
       select(choose(choose(count > 8, |
          4 - (count - 1) % 4, |
          (count - 1) % 4 + 1), |
          ?Tab1, ?Tab2, ?Tab3, ?Tab4))
       ?sheet{choose(int((count - 1) / 4) + 1, |
          prop:above, prop:right, prop:below, |
          prop:left)} = true
       ?sheet{choose(int((count - 1) / 4) + 1, |
          prop:angle, prop:down, prop:upsidedown,|
          prop:up)} = choose(int((count - 1) / 4))
       count = (count % 16) + 1
     end
   end
   close(window)
```

If you take out the line breaks necessary to format Gordon's solution to fit in this space, it amounts to only ten lines of code! And it works perfectly.

My thanks to Carl, Brice and Gordon for taking the time to contribute. I've learned a few things about the property syntax, and I've been reminded that CHOOSE is a remarkably useful language statement.

Download the code

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

Introduction To SQL: Part 3

by Dave Harms

Published 2001-03-14

Part 3

If you've been following along in this series of articles you'll be familiar with some of the pros and cons of switching to SQL (well, mostly pros – the cons are definitely less plentiful) and you'll have an idea of how to go about creating one or more SQL databases. Of course databases are nothing without data, so in this installment I'll look at some of the ins and outs of creating SQL tables.

Creating tables

As you'll recall from part 2, the SQL language isn't just for retrieving and updating data; you also use SQL to create databases, tables and indexes. Here's a simple SQL statement that creates a table with a few fields:

```
CREATE TABLE  ArticleGroups
(
   ArticleGroupID   INT AUTO_INCREMENT NOT NULL,
   Title            VARCHAR(255),
   PublicationDate  DATE,
   Value            TINYINT,
   LastModified     TIMESTAMP
);
```

For the sake of readability, I've put each field (or to use the SQL term, column) definition on its own line. This is common practice as it makes the statement more readable.

You will have to get used to some different data types when you switch to SQL. For a complete listing of available data types, start with the ODBC driver help, ODBC Data Types section in the Clarion online help. You'll see a chart mapping ODBC data types to Clarion data types. Some of the most common type comparisons are as follows:

ODBC Data Type	Clarion Data Type
CHAR (fixes length string)	STRING
VARCHAR (variable length string)	CSTRING
DECIMAL	PDECIMAL
TINYINT	BYTE
SMALLINT	SHORT
INTEGER	LONG
DATE	DATE
TIME	TIME
DATETIME	GROUP (see below)
TIMESTAMP	GROUP (see below)

Conversion between data types isn't always straightforward, and there are often alternatives. For instance, you can represent an ODBC fixed length CHAR string with either a Clarion STRING or a CSTRING, but if you use a CSTRING you won't be able to use Clarion to create that table. Or perhaps your design calls for a BYTE field but your SQL database doesn't have a TINYINT type. In that case you'll need to use a SMALLINT or other data type. You can also represent many SQL data types as a Clarion STRING.

Dates and times

One of the more significant differences in data types between TPS and SQL tables has to do with date and time handling. If you're a long-time Clarion developer, then you're familiar with storing dates and times as LONG values, and displaying these values with pictures. A Clarion standard date is the number of days since December 28, 1800, so March 13, 2001 is 73124.

If you're viewing a TPS (or DAT) file with a Clarion program or utility, you can specify an appropriate date picture to translate the date number into something meaningful. But if you're using a program that doesn't understand a Clarion standard date (say you're browsing a TPS table in Excel, using the ODBC driver), then you're just going to have a bunch of numbers to look at. Quick, what day is 73087? And Clarion standard time is no better, since it's stored as the number of hundreds of a second since midnight, plus one.

In a mixed environment, you need to store dates and times in a format non-Clarion programs can understand. That's the reason

for the DATE and TIME types. Both of these are four byte variables. DATE uses two bytes to store the year (1-9999), one byte for the month (1-12), and one byte for the day of the month (1-31). Similarly, TIME uses one byte each for the hour (0-23), minute (0-59), second (0-59), and hundredth of a second (0-99). You don't have to do anything special in your code to use a DATE or TIME instead of a LONG; in fact, if you're working with TPS tables you can change all of your LONG fields to DATE or TIME fields, and just convert the files in the dictionary browser. Now you're one step closer to being able to switch over from TPS to SQL.

Not all SQL databases support individual DATE and TIME data types, but most have a DATETIME or TIMESTAMP data type, which is a combination of a DATE and a TIME. These two data types have different purposes. A DATETIME is usually set to a previously known value, while a TIMESTAMP is automatically set to the date/time the record was last modified. In some SQL (but not all) databases TIMESTAMP is guaranteed to be a unique identifier, although this requirement is not part of the ANSI SQL specification.

Clarion doesn't have a data type to correspond to the combination of date and time, but it's easy to fake. When you import a SQL table definition which contains such a field, you'll see a group structure in the table definition that looks something like this:

```
TimeStampField    STRING(8),NAME('TimeStampField')
TimeStampGroup    GROUP,OVER(TimeStampField)
TimeStampDate       DATE
TimeStampTime       TIME
                  END
```

In this example the date/time field in the SQL table is called TimeStampField, and it's declared as a STRING(8). The fact that TimeStampField is a STRING isn't important. What is critical that this field be eight bytes long, because that's the length of a date/time field in SQL. The second part of the declaration is a GROUP with an OVER attribute. OVER lets you assign one variable to the memory space occupied by another variable. When you request data from this table, and the driver returns the contents of TimeStampField, you can refer to those contents eight bytes at a time using the TimeStampField label or the TimeStampGroup label. In either case you'll get gibberish. But if you use TimeStampDate or TimeStampTime, you'll be working with just the four bytes corresponding to that half of the variable. The OVERed GROUP is just a trick to extract the individual DATE and TIME values.

If your SQL server doesn't support individual DATE or TIME data types, and all you need is one or the other, you'll have to use a DATETIME (which, confusingly, is sometimes just called DATE) and resign yourself to wasting the four unused bytes.

Primary keys

Take another look at that CREATE TABLE statement from the beginning of the article:

```
CREATE TABLE  ArticleGroups
(
   ArticleGroupID   INT AUTO_INCREMENT NOT NULL,
   Title            VARCHAR(255),
   PublicationDate  DATE,
   Value            TINYINT,
   LastModified     TIMESTAMP
);
```

Do you notice anything missing? There are no keys in this table definition. And this is a critical omission, because every table should have at least one key, called a primary key.

Any database system needs a way of uniquely identifying individual records within the database. In a flat file database, this record pointer is often the byte offset in the physical data file of the start of that record. That's the case with TPS tables; with Clarion DAT files the pointer (as returned by the POINTER() function) is the actual record number, which is why in DAT files you can use code like GET(Control,2) to get the second record in a file.

SQL databases don't necessarily have a built-in record identifier, which is why you have to be sure to provide a way of uniquely identifying each record. Here I've modified the ArticleGroups table create script to include a primary key on the ArticleGroupID field.

```
CREATE TABLE  ArticleGroups
(
   ArticleGroupID   INT AUTO_INCREMENT NOT NULL,
   Title            VARCHAR(255),
   PublicationDate  DATE,
   Value            TINYINT,
   LastModified     TIMESTAMP,
   PRIMARY KEY (ArticleGroupID)
);
```

Also note that in this definition (which is from a MySQL database) I've added an AUTO_INCREMENT attribute and a NOT NULL attribute. Every time I add a record to ArticleGroups, the server will assign a unique, automatically incremented value to ArticleGroupID. The NOT NULL attribute means that ArticleGroupID must have a value; if I didn't use the AUTO_INCREMENT feature, I'd have to supply my own value for ArticleGroupID.

Keys and indexes

Clarion developers have had to deal with some confusing terminology in recent years. Is that an object or a class? A table or a file? And now, is that a key or an index?

In SQL parlance, a key is a logical definition, not a physical thing. Keys are how you relate two tables. In the Clarion Magazine database I have a table for groups of articles (ArticleGroups), a table for articles (Articles), and a linking table that manages the many-to-many relationship between ArticleGroups and Articles. This table is called ArtGrpLink, and its create statement looks like this:

```
CREATE TABLE  ArtGrpLink
(
   ArtGrpLinkID    INT   AUTO_INCREMENT NOT NULL,
   ArticleGroupID  INT   NOT NULL,
   ArticleID       INT   NOT NULL,
   LastModified    TIMESTAMP,
   PRIMARY KEY (ArtGrpLinkID)
);
```

In ArtGrpLink, the ArticleGroupID field's only purpose is to link a record in that table back to the Articles table. ArticleGroupID is what's called a foreign key – it's a key because it is used to link to another table, and it's foreign because the information it links to exists outside the current table. *But there is no separate physical structure that defines the key, as there is in a TPS table.*

When you create a TPS table, and you want to sort that file in a particular way, you define...a key. There is something comparable to a TPS key in SQL databases but a) it's called an index, not a key, and b) it's optional.

This may take a bit of thought. A key, in SQL, is just one or more fields that link to another table. When you define a key (I'll give an example shortly) you tell the server that a relationship exists between two tables. Optionally you may also tell the server that there are certain constraints on this relationship. Perhaps when the parent record is deleted you want to delete all the child records automatically, or you want to prevent the parent from being deleted if child records exist.

Is this starting to sound familiar? In SQL, defining keys is akin to defining relationships in the dictionary editor. On one side of a relationship you have a foreign key, and on the other side you have a primary key. Here's an example of an employee table with a foreign key (dept_id) that references the department table's primary key (also labeled dept_id).

```
create table employee
(
    emp_id            integer       not null,
    manager_id        integer                ,
    emp_fname         char(20)      not null,
    emp_lname         char(20)      not null,
    dept_id           integer       not null,
    primary key (emp_id),
```

```
    foreign key ky_dept_id (dept_id)
      references department (dept_id)
      on update restrict
      on delete restrict
);
```

This foreign key declaration says that `employee.dept_id` matches `department.dept_id`, and that the server should not allow anyone to delete a `department` record or change the `department` record's primary key (`dept_id`) if there's a related `employee` record.

This particular foreign key example only restricts updates and deletes, but you can also have foreign keys with no constraint, a cascade constraint (delete or update all records with a matching foreign key), or a constraint that doesn't change the related record but clears the foreign key or resets it to a default value if the primary key record is deleted.

SQL indexes

Just as SQL keys correspond to Clarion data dictionary relationships, so do SQL indexes correspond to TPS keys. In a TPS table a key is a physical construct that makes it possible to sort records by field contents. In a SQL server, an index is a physical construct that lets the server sort records efficiently.

Technically, TPS keys and SQL indexes are both optional, since Clarion applications use the View engine to collect data, and the View engine lets you sort on arbitrarily chosen fields. If you don't have a key/index defined, performance will suffer to some degree, probably more with the TPS table than with the SQL table. And most of us who use TPS files wouldn't think of creating a TPS file without at least a few keys.

When it comes to SQL, however, the whole subject of creating indexes (which, of course, correspond to TPS keys) becomes a little more nebulous. That's because an application that wants to retrieve some data from a SQL database simply sends SQL statements like:

```
SELECT Name,Address,City,State FROM Names
  ORDER BY State;
```

The `ORDER BY` clause tells the server what order to use when sorting the resulting data, but there is no instruction on which index, if any, the server should examine for that order. Now, with some SQL databases you can give the server hints in situations where several indexes could be used, and you know one to be more efficient than another, but that's beyond the scope of this article. For the most part, how the server does its job is up to the server.

This abstraction of the query language from the server implementation has important consequences for the developer.

With TPS files, the Clarion runtime library ensures that the definition you have for your file exactly matches the file itself; in SQL, all that matters is that the fields you have defined in the table have corresponding fields in the table on the server. Your definition may have only one field, while the actual table may have dozens.

You also don't need to define any indexes for your SQL tables in the data dictionary (okay, the dictionary editor calls these keys – life *can* be confusing). You can create a browse that displays in record order and add whatever additional sort fields you like, and if the server has corresponding indexes, you'll get good performance.

You probably will want to include index (key) definitions in your SQL tables in the dictionary, however. For one thing, you'll need indexes/keys to define relationships between tables, and for another it's a good thing if your data dictionary fairly closely models the actual database. But there's no requirement for your indexes to match the back end's indexes; you'll never get an invalid record declaration because of an index mismatch. You might get some abysmal performance because your code assumes an index that doesn't exist, but that's another story.

Summary

If you've never worked with SQL tables before, you'll first encounter some unfamiliar data types. Most of these have close Clarion equivalents, the exception being date/time combinations, which require a special GROUP structure. You'll also need to ensure that all your tables have a primary key, which is a field or combination of fields guaranteed to be unique for each record.

For the most part, SQL tables aren't that different from TPS tables, at least until you come to the subject of keys and indexes. It can be tricky at first remembering that a foreign key defines a relationship between SQL tables, and an index is a physical construct which helps the server locate records more quickly. As well, because your SQL application works with data by using SQL statements, not by directly accessing data files, there can be significant discrepancies between the table and index definitions in your applications and those on the server, and your application can still work successfully.

When this series resumes next month I'll begin creating a small SQL application from scratch. In the meantime, if there are any aspects of SQL application development you'd like me to cover for this series, please email me at dharms@clarionmag.com.

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

Clarion MAGAZINE

Search
Go

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

Feature Interview: SoftVelocity's Bob Zaunere

by Dave Harms

Published 2001-03-12

Bob Zaunere, SoftVelocity's President and CEO, recently spoke with Dave Harms about the Clarion product line and SoftVelocity's plans for the future.

You shipped Clarion 5.5 last November. What has the response been like?

Clarion 5.5 has been really well received. We get a lot of email from people who are really happy with the stability of 5.5, and who are pleased that the product they'd waited so long for was actually delivered. There were areas that we just couldn't address that we had planned to address, but most of them we were able to get to.

It was a long wait between 5.0 and 5.5.

TopSpeed was ready to release 5.5 just about the time of the transition [of Clarion ownership to SoftVelocity]. I held it for a lot longer than most people thought I should, including a lot of the Development Centre. I was catching a lot of flak from people who said "Just ship it." I said no, we'll ship it when we get enough feedback that it's really stable. And it shipped as a very good, solid product.

Is the time between 5.5 and 5.5a about what you expected?

I think so. We started shipping 5.5 at the beginning of November, and then before you know it you have the holidays and that affected everybody, so essentially we had about three months to turn this release around. In that time we weren't strictly just working on bug fixes. We've got to continue to develop new projects as well as enhancements, some that began as long as two years ago, so I'm pretty happy with the delivery of the first release in that timeframe.

Do you plan to release no-cost updates on a regular basis?

That's part of the strategy. Things that need fixing will be fixed,

and there will be no charge for that. If we start talking about enhancements, they'll either be delivered as a full version release (5.5 to 6) or as a dot release. I lean more towards saying if we can come out with major maintenance releases on a three month cycle, or sooner, depending on the scope of what we want to fix, and come out with dot releases two per year, and maybe a full version release every year, I'd be pretty happy.

You've been getting some heavy hits on the 5.5a update. Do you have enough capacity to handle the downloads?

We host with a company down in Miami. We were really pleased with their performance, but they recently did a major push to get market share in Latin America, and they added 10,000 new customers. So I think we may be looking to host with another [company]. They claim to have gigabytes of bandwidth, but seeing is believing.

Is all that traffic an indicator of Clarion sales?

New license sales have continued to both make us happy and amaze us. We sell quite a few new licenses per month, and we expect that trend to continue.

How's the "sell with them" approach to marketing going?

It's worked really well, and that's reflected in our sale of new licenses. We have some work to do in that area, and what we plan to do is to offer some free web seminars that will introduce people to the product and what you can do with it, and just make the seminars open to anybody. Short little things, maybe twenty minutes or a half hour, either on-line or downloadable, and [we'll] let them get distributed.

One of the new features you have in the works is more direct support for SQL. What will this look like?

The basic premise for the SQL product is to directly support the use of the SQL language, static or dynamic, using PROP:SQL for all queries, updates, and deletes. The advantage we see with this is the resulting code is very easy to read and extremely easy to optimize.

Is this a rewrite of the ABC browse?

It is entirely separate. It is not an extension or add-on [to the ABC browse]. It is strictly its own beast. The reason behind it is twofold. One is that the code for a browse is not trivial, to say the least. It doesn't have to be [that complex]. If you can take away some of the code and put the logic onto the backend, you can really simplify what the code looks like. The other reason is performance. Our driver technology can really perform well given the right circumstances. We have more and more users who are moving toward SQL, and ABC doesn't always deliver the

performance they want, particularly in larger operations where there's a lot of concurrent users.

[Another] area that we're looking at is support for stored procedures. We want to be able to generate code to call existing stored procedures, whoever may have written them. Now creating the code that maps stored procedure parameters correctly can be fairly time consuming. Every variable has to be correctly mapped for the call to the procedure to work. Not everyone has the expertise to do it correctly. If you can import existing stored procedures and then get Clarion to generate code to call those stored procedures correctly, you save yourself a lot of work. And if you add to that the ability to generate new stored procedures based on the metadata in the dictionary, you can really accomplish something useful. And that's where we're going. The premise that we have is to provide a really simple, extensible interface to get optimal performance.

You've also announced an ASP product. How does this fit in with your web development strategy?

Pure web apps and apps that run over HTTP [on a local machine] are a separate subject. A pure web app to me is one that uses one of the standards for web applications, like ASP, JSP, Perl, and the like. And that's obviously what we're targeting with the ASP project. ASP is the next logical choice for us. It happens that I like JSP a little bit better than I like ASP, but ASP is supported everywhere. And that's really what we want to see happen, to give Clarion developers applications that they can host anywhere they like.

The mixed development is not a high priority any more?

No. The ASP work will lend itself to being able to call into business logic stored in a Clarion EXE or DLL, but that isn't the focus of it. There are things that ASP applications are really good at, and this is what we can deliver to you.

So the app broker is going to be orphaned?

Not at all. I probably should time this with the press release, but I'm sure you know the ClarioNet product? We have an agreement to become the exclusive distributor for it, so we're going to tweak the app broker a bit. It's a much better use for the app broker than the current approach of generating HTML.

So the web development will split off into two product lines, one for ClarioNet and one for ASP and similar code?

We will maintain and make improvements to the Web builder technology – like the HTML web reports we added in 5.5a, and make improvements to the Application Broker itself. A web builder application can be integrated with ASP [or similar code] to deliver business processes from within a Clarion app to a Browser, processes that would be very hard to model and code in VBScript.

The Web Builder technology and ClarioNet are very closely linked because they share the same architecture. But, one of the things that I've felt for many years we didn't do a good job of at TopSpeed was take advantage of the code generator and the metadata in the dictionary. I want to change that. The ASP code generation is a small step in that direction. The work being done to accomplish that will lend it self to, say, JSP. It won't be that big of a step. Or we can do PHP or Perl. PHP will probably be a better bet for us [than Perl]. It will differentiate the product. You'll have the ability to go and choose a language based upon your preference or your knowledge of a language, and its unique abilities. Essentially you get the same functionality with all of them.

Would you be able to switch between templates to move an application from one language to another?

That I can't determine yet. Conceptually it seems like you should be able to do it, but until we go down that path a little further it's hard to say with any certainty.

How is the ASP project coming along?

Fantastic. There's so much we can do with the metadata in the dictionary and the AppGen code generator. ASP code can be difficult to write and even more difficult to read and maintain, mostly because it's so tedious. You're back to keeping all that [information] up in your head. All of the things we did for 4GL programming, we can apply the same principles to ASP. We've got many features already completed, common requirements like page level security, data lists, update forms, database i/o error checking and recovery, and a utility library for things like formatting result sets, populating drop lists, search capability, user-sortable tables, that type of thing.

There are two approaches to ASP. What we're doing right now, all of the applications are server-side VBScript. The VBScript will execute an SQL statement, get the result set, and generate the HTML to display the page to the user. The primary advantage of that approach is that a lot of ISPs don't allow the registration of any server-side ActiveX DLLs or ISAPI extension DLLs, the app broker being an example. And even those ISPs that do support customer-provided DLLs often have special charges for installing them, or they require customers to move to a dedicated server plan, which can cost a whole lot more money.

The disadvantage [of using VBScript this way] is that you don't have a separation of presentation and business logic. The second phase of the project that we could deliver would separate that out, by taking the business logic and putting it into a COM object, [similar to the JSP/custom tags approach].

Are you planning any XML support?

Absolutely. The XML solution is going to give you the capability to read and write to IE5 XML data islands, and to ADO persisted XML

or to XML files with XML schemas. Essentially what will be different about our approach is that it gives Clarion application the ability to read and write XML using standard ANSI SQL92 statements. That includes support for SELECT, INSERT, ORDER BY, GROUP BY, etc. For our development community, people who have a database slant to their experience, it's going to be especially attractive. Essentially it's just another driver at that point. A user would be able to set up a local data source that contains a URL to an XML file anywhere on the web or on a local resource, and read or write to it.

Which parser will you recommend, or will you supply one?

We're definitely not going to include a parser. The Microsoft XML parser or Xerces are two good choices. If you want to go beyond [the XML driver], we'll provide a wrapper class for probably both those two, but at least the Microsoft parser. We also have someone looking at what it would take to wrap up the Xerces parser.

Any thoughts about adding SOAP capability to Clarion?

Nothing I want to discuss in detail yet. We've looked at SOAP, and there are some aspects that are very attractive. When we looked at how we could do it, we came up with what seems like a fairly easy approach to implement, using our XML driver as a component on the server. Part of the solution is tied in with the XML driver, the other part is a Clarion application on the desktop. The answer is probably [won't be] complete in the next couple of months; it won't come out as an official implementation at the same time as the initial XML support.

Which suggests the XML support will be out in the next couple of months.

Yes, Absolutely.

How about the 32 bit IDE?

A lot of the work to get the environment to 32 bit was done as part of the IBuild project. Most in fact was done.

This was just a straight port, no new functionality?

Yep. Some of the things that weren't ported, like the editor, are easy to replace or even rewrite. Fairly easy, but not trivial. And some things have been around for so long that nobody wanted to touch them initially, like the window formatter and report formatter, both of which could stand a complete rewrite. The areas that aren't in 32 bit yet are areas that we're not sure we want to put into 32 bit as they stand today.

Will there be a 32 bit IDE in C6?

I think the community has pretty well decided that if we release anything with the label C6 that isn't 32 bit, we'll be run out of town. So the answer is there will be a C6, and it will be 32 bit. The 32 bit IDE work does continue. What we've really focused on is rewriting the sections of code that affect robustness and stability. There are more than a few people who are well respected and who work with us who think we ought to be looking at 64 bit. Right now we're hybrid 16/32 bit, both in the environment, which isn't as significant except where it impacts COM support as in the formatters. The fact that we are 16/32 always gave us a nice advantage. What we're looking at now is what if we leapfrog to 32/64.

Is there a pressing need for 64 bit performance, or is it more of a marketing issue?

It's more to keep pace with the direction the industry is going in. There will be performance advantages, but with 1 GHz machines for under, I think, $800, performance is less of an issue.

What's the status of COM support, like the COM interface Jim Kane wrote about recently in ClarionMag?

The COM interface rewrite was part of a project, started well before 5.5, targeted at providing COM syntax almost completely like VB's, and improving the interface to COM objects as well. It's undocumented because it's incomplete. That work hasn't been a priority in the last 3 months...

The dot syntax isn't there yet.

In order to provide [VB COM syntax] and not hybridize the language, we need to move the entire language to full and complete support for dot syntax. When that's done, we'll be where we want to be.

Is dot syntax a C6 feature?

Probably prior. We can incrementally release updates that move in that direction. What we just released in 5.5a were some significant bug fixes. There are many more fixes that are internal, core fixes and enhancements that wouldn't make sense to anybody if we documented those [publicly]. Simultaneously we have new development going on. What we have to decide is do we hold back until everything that we'd like to see in C6 is done, or do we deliver things incrementally. I'm of the opinion that delivering technology incrementally is a better approach to take. The company does better and the developers get their hands on the technology quicker, and everybody's happier in the end.

Do you have any plans for ADO?

We sure do. We started to work on ADO four months ago, or a little longer. Recently Andy Ireland, who worked in the Development Centre, and now works in the U.S., offered us the

considerable amount of work that he had done for his own project. We'll be combining the two works, working closely with Andy, and we'll write templates that wrap Andy's classes, and we're going to be adding additional driver support.

Do you expect to have support for native threads in C6?

Yes, if not before. I don't know if we'll hold that back for C6. Three months or so ago I put rewriting the thread support for the 32 bit RTL at the top of the list. That [makes it possible] to get rid of data swapping completely. Even though the documentation says the RTL allocates memory for threaded data instances for every thread, in the current implementation if you use the ADDRESS function on a threaded variable, it's going to be the same [address] on all threads. [The RTL swaps] instances in and out when the thread becomes active. This is one of the reasons for GPFs and memory corruption, and thread synchronization errors. If we exclude swapping, threads can be synchronized very easily. The downside is that the ABC classes use the fact the address of the variable is essentially the same across all threads. It's possible, of course, to change those classes, and the problem then is that changing classes can break programs that depend on those classes.

And there's a whole other area of synchronization and true processes that Clarion developers will need to become familiar with.

That's true. I think the initial implementation was very clever, but it was shortsighted. Eventually it came back to haunt us. The positives if we make the changes is that you'll get much more robust code. And faster execution, pre-emptive task switching, a whole slew of benefits. The only downside is the potential for breaking existing programs.

You mentioned JSP earlier. Any other plans for Java?

We actually have a research project going to do client side or desktop Java. It seems quite reasonable to take advantage of Clarion's core strengths with the use of metadata and code generation. We're looking at the current JDK to see what we'd need to accomplish to deliver a Java client program. I think it's just a natural. I tried to convince Bruce [Barrington] to do this when the first JDK was out maybe two, three months. I wrote a couple of programs, I said "Hey, Bruce, look at this, this is going to be hot stuff! People are foaming at the mouth over [Java], and we can turn them into Java programmers overnight." And I was shot down on that one. It was bytecode. "That's pseudocode! That's where we came from! We're not going back there..."[laughter] Of course we'd paid a heavy price to move from pseudocode, so I could see where he was coming from.

Client side Java does seem to be making a bit of a comeback.

I think it is. As the performance of CPUs continues to go up the performance of Java becomes less of an issue. For well over a year I've been reading that Java is pretty much equivalent to C++ in many benchmarks. For us, Java will get us into places that we can't get into otherwise. In a lot of companies, if you don't say Java, they don't even want to talk to you.

How do you deal with some of the language differences, such as how Java lays out controls on a window using layout managers as opposed to fixed positioning?

The people we've had look at this, and we've actually had some people who have been users who have now been programming in Java for some pretty well-known corporations, and I brought up that exact issue to them. And we were actually approached by two separate corporations who said "Look, we really like Clarion, but we need Java, and we'd like to help you get there." Code generation is something we're great at, and I brought up the exact point that you brought up. There are really only two ways to handle it. One is to say okay, you lay it out and we make a best guess of what you intended. And that isn't entirely bad for Java, but you still don't get to see what you would have done until you actually run the program, which maybe isn't ideal. The second thing, which we could do, is to provide a plug-in, and the plug-in would be a Java module that would render up the UI and show it to you. The final step would be to render it up into a formatter and let developers adjust the design and save it back, and now you have a complete formatter.

I've heard from people who say don't even worry about that, [who have] given up on positioning in Java, literally. And these are guys from shops that are pure Java. So maybe it won't be an issue. In my own mind, I'd rather see us say "That's fine, but we're going to at least let you see what it looks like." And depending on the difficulty, and what the time line and the costs would be, maybe decide we can do a little bit better than that. We can at least let you render it and change layout schemes and layout managers and fool around a bit with color and sizes. Time will tell how far we go down that path.

Do you foresee an alpha available any time soon?

It's still in the research stage.

Is Java where you'd be looking for your cross-platform capability? A lot of developers have been asking about a Linux version of Clarion.

We're not currently looking at a Linux version of Clarion, but as Linux evolves on the desktop that may be something that becomes more important to us. From my experience, Linux is strongest on the server side, particularly on our continent. Even [in Europe] I think it's real strength is on the server side. To port the development tool over, at this time I can't say there's a push to do that. Now to run a Clarion application on a Linux box, we could

accomplish that with the ability to create a Java client, or a C++ program.

How about C#?

We've been looking closely at the whole .NET framework. There's a lot of overlap between a requirement to generate Java and a requirement to generate C#.

The specifications for those languages are very close. It seems that a developer could move from one to the other quite easily.

No question there is a great deal of common ground. The same principle holds as for JSP/ASP. Right now Java obviously has the mind share for developers and corporations. C# is increasing, but I've looked at some surveys that say a lot of developers aren't interested in C#, and surprisingly so. People who built their careers on C++ aren't jumping on the bandwagon, people who do Java definitely are not interested.

Any plans for Windows CE or other handheld devices?

The first interview we did I talked about the untapped possibilities of our code generator, and I think that applies to Windows CE and to handheld computing in general. There isn't an immediate link between a Clarion application as we traditionally know it, and a handheld device. There is an inherent capability of the product to say we can automate a large portion of the requirements for a Win CE app. Do we have anything that we're doing today? No. We looked at it, [asked] what would it bring to us as a company. Right now it doesn't have the appeal of other things we can do, but it could become a part of the future using the basis of other work we are focused on.

How do you best apply your resources to stay competitive in the development tool marketplace?

In a lot of ways I think we're more able to respond to the market than Topspeed was. At any one time now we'll have anywhere from two to five research projects running, with one to four developers just looking at what it would take to integrate, add on, or extend Clarion. Most of these developers have been contracted with because they have expertise in the technology that we're interested in. We then get the ability to access the very best people who have expertise in something that we're interested in, without having to train people internally and the ramp-up time associated with that just to investigate feasibility. There isn't anything that's impossible to do if somebody with talent and experience puts their mind to it. What we need to do is identify what areas we want to go in. In the past the development staff was pretty good sized, we had a team of guys that worked on the same technology in the same office, and shared the same vision of where they all should go. What we've been able to do is find people who aren't connected to the Clarion community at all, and

say [to them] "Look, we're interested in this particular technology, and this is what we want you to do." We take the results and we decide if it's something feasible for us. For instance, that's what we're doing with the Java client side project.

You're really branching out into other languages in a big way.

Absolutely. What's really becoming clear is that Clarion can easily evolve into a language-independent product. Yes, we have a core language, and yes we have incredible capabilities in the language and a fantastic compiler and linker , but if you want to use Clarion the development tool you won't be restricted to just the Clarion language.

How about the CBT courses? Anything new?

Yes. One of the courses that we're going to do is in conjunction with the SQL project. The reason being that through firsthand experience we find that a lot of people want to move to SQL and they need two things. They need to learn a little bit about the differences between ISAM file processing and SQL processing. And two, they need to learn the specific implementation of Clarion and how to best optimize it. So that is one that we're going to do. We'll do a course covering the XML support, without question. We may or may not do one with ASP, and the reason is just that there's so much information about ASP out there that it might not be that useful. On the other hand we end up saying okay, here's either a little mini course you can download or something that comes as a CD and just shows you what we can do, and if you really want to learn ASP, just go and pick one of the four dozen books on the market and I don't know how many videos and CBTs that deal just with ASP.

So you're shifting your focus to CBT and away from classroom training?

Not really, we still see great value in classroom training. What we found is that unless we really start reaching out across the country a lot of people either cannot get the time off, or can't afford the cost of travel plus hotel plus the course. So rather than hold them back, the idea of letting them take the course either online or via CD is pretty attractive. As far as the instructor-led training, we're routinely running classes. We try to run at least two every month, and we are looking at doing classes outside of Florida.

Will there be a DevCon 2001?

The answer is yes, but we haven't decided exactly when and where. I won't do anything along the lines of the Pier 66 conference, and mostly because of the cost. I thought it was overpriced for many developers who wanted to attend. I want to do it so it's more affordable to developers, and so that it actually can generate some revenue for the company. And the other thing I will do, without question, is put 98% of the focus on technology

and technology transfer, and 2% on marketing. We could make a dot release, and say 'attend the conference, get a dot release, and here's what you'll learn'. Make the whole thing essentially training courses. Including of course presentations and demonstrations of technology that won't be ready to ship.

What are you interests and hobbies outside of Clarion?

Is there anything outside of Clarion? I'm kidding. With two kids, a lot of my hobbies have changed dramatically. No more sky diving, no more hang gliding, a lot of the things I enjoyed have been vetoed. I used to be an avid sky diver, and when I could get to somewhere that wasn't flat I liked to do some hang gliding, which you can't do much of down here. I did some ultralight flying, and that was pretty enjoyable. I used to like a lot of the air sports. Kelly vetoed all that. But I said when the kids get older, though, I'm going back to it[laughter].

Clarion Magazine Readers Respond

Following the publication of this article we asked Clarion Magazine readers to respond. Here are the results of the poll as of March 21, 2001:

The response numbers were as follows:

Completely	75
Mostly	142
Somewhat	22
Not at all	6

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

Clarion MAGAZINE

Search
Go

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

Introduction To SQL: Part 2

by Dave Harms

Published 2001-03-06

Part 2

In Part 1 of this series on SQL databases I explained some of the important advantages of SQL databases over flat file databases (such as those you create using TPS files). I'll assume that you don't need any more convincing, and you've decided that you really do want to move forward with SQL. In this installment I'll cover the basic differences between creating a SQL database and a TPS database.

To use Clarion and SQL, you need two things: a SQL database, and a driver that lets Clarion talk to that database.

Choosing a driver

Clarion 5.x Professional ships with the ODBC and MSSQL drivers; Clarion 5.x Enterprise adds the SQL Anywhere (a.k.a. Sybase) and Oracle Accelerator drivers. If you only have the Professional version you can still use Oracle, SQL Anywhere, and many other SQL databases using the ODBC driver.

What you won't get with ODBC is the ability to do multi-file imports from the SQL database, and you won't be able to use the Clarion Database Synchronizer to transfer changes from the database to your dictionary and vice versa. I'll come back to the Synchronizer in a later article. For now, I'll assume that you're using the ODBC driver (which is still a fine solution) to talk to your SQL database.

Choosing a SQL database

It's possible that your choice of SQL database has already been made for you. If this is the case, chances are you're looking at one of the commercial SQL servers, such as Microsoft SQL Server, Oracle, or SQL Anywhere (Sybase). There are also an increasing number of good, free, open source servers available, including MySQL, PostgresSQL, and Interbase.

Each SQL server has its own strengths and weaknesses. For many developers, commercial database licensing costs are a serious drawback, but commercial products may have features not available

elsewhere. And although all SQL servers share a common core of SQL statements, most vendors have enhanced SQL with database-specific commands. If your applications begin to make significant use of server-specific functionality, you may find it difficult or even impossible to port your application to another server.

For example, I use MySQL as the core database behind Clarion Magazine, in large part because of MySQL's reputation for speed and reliability. But MySQL's support for transactions is new and relatively untried. If I needed transactions I might think seriously about PostgreSQL (a.k.a. Postgres). Postgres is considered fairly reliable, but doesn't deliver data as fast as MySQL does, and has an 8KB record size limit, which I find a bit too restrictive. On the other hand, Postgres has sub-selects (you can use one SELECT statement as part of another SELECT statement) which I'd dearly love to have in MySQL. I can work around the missing sub-select capability with temporary tables, but that's a bit awkward. On the whole, I think MySQL is a better choice for this particular application, although I've read that Postgres may actually scale better...

You get the idea. Throw in a few more key features like triggers, stored procedures, the ability to store programming objects in the database, and you have a whole lot to consider when choosing a SQL server. Fortunately, there's a lot of competition for SQL customers, so you can usually get a low-cost or free trial version of just about any database server you want to take a look at.

Installing the server

Whichever server you decide on, your next job is to install the server software. In a production environment you'll almost always install the database server software on a dedicated computer (also referred to as a database server). But for development work you can often install a personal version of the server on your own computer. Although you won't get a realistic assessment of the performance your application can expect in a networked installation (depending on the computing and network hardware involved, an application may run faster or slower using a personal server installation than using a networked database server), you'll still be able to test all of your application's functionality.

Creating a database

You might think that one of the advantages of running a personal SQL server on your own machine is that you won't mess up somebody else's database. In fact, SQL servers let you create multiple databases for various purposes, so it's easy to isolate your test or development data from other data on the server.

A SQL database is a bit like a directory of TPS files. Just as you can have same-named TPS files in different directories, so you can have same-named tables in different databases. How the SQL server actually stores the databases is entirely dependent on the server implementation, but that's a detail you don't need to concern yourself with. In SQL, you deal with table definitions and data; you let the server worry about where and how it actually stores that data.

A SQL server can contain a large number of databases, and many tables within each database. The SQL server can also actively manage all of this data to ensure efficiency and integrity. Larger organizations, with large databases, typically employ one or more Database Administrators (DBAs)to ensure that nothing bad happens to all that important data. To make the DBA's job a little easier, most SQL servers allow the DBA to set various permissions on different table operations. At the lowest level, a particular user may only be allowed to view a particular column of data in a particular table in a particular database. At the highest level, a user (really now a DBA) can create and delete entire databases.

If you're working in a larger organization, and the departmental SQL server is your only option, the best you can hope for is that the DBA will give you your own database in which you can create and delete tables to your heart's content. If you have your own SQL server, then you can do anything you want, can't you? Ah, the power.

There are two ways to create a database. One is to execute a `CREATE DATABASE` statement. The other is to use a database administrator utility which provides a somewhat friendlier interface, but which will ultimately execute that same `CREATE DATABASE` statement.

Creating a database with SQL

To create a database called Test, you use a SQL statement like the following:

```
CREATE DATABASE Test;
```

Notice that the statement ends with a semicolon, which in SQL is the line terminator. You can spread a SQL statement over multiple lines by pressing the Enter key, which in the case of longer SQL statements often greatly improves readability. The SQL server won't attempt to execute the statement until it encounters the terminating semicolon.

Creating a database is simple, isn't it? But where do you type this command? SQL servers generally come with a number of bundled utilities, one of which will be a SQL interpreter, sometimes called a SQL query tool, or SQL query analyzer. You can use this tool to execute any valid SQL statement (at least one you have permission to execute). For example, I have MySQL running on a Linux box on my office network. To interactively execute SQL statements on that server, I telnet to the Linux box and type mysql at the shell prompt to run the mysql client program. Figure 1 shows how I use that program to create a database called APlaceForMyStuff.

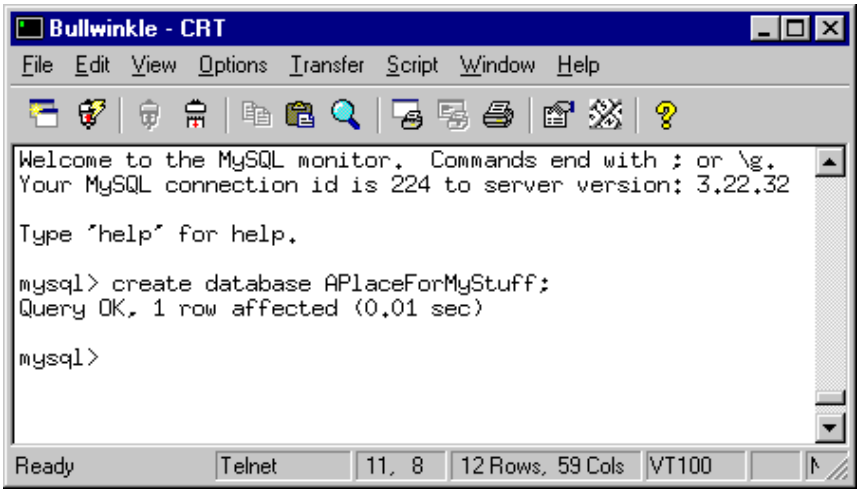


Figure 1. Creating a database with a direct SQL statement

Creating a database with a database administration utility

It's probably more likely that you'll create your databases using an administration utility. Figure 2 shows Sybase Central , the database administration program that ships with the SQL Anywhere database.

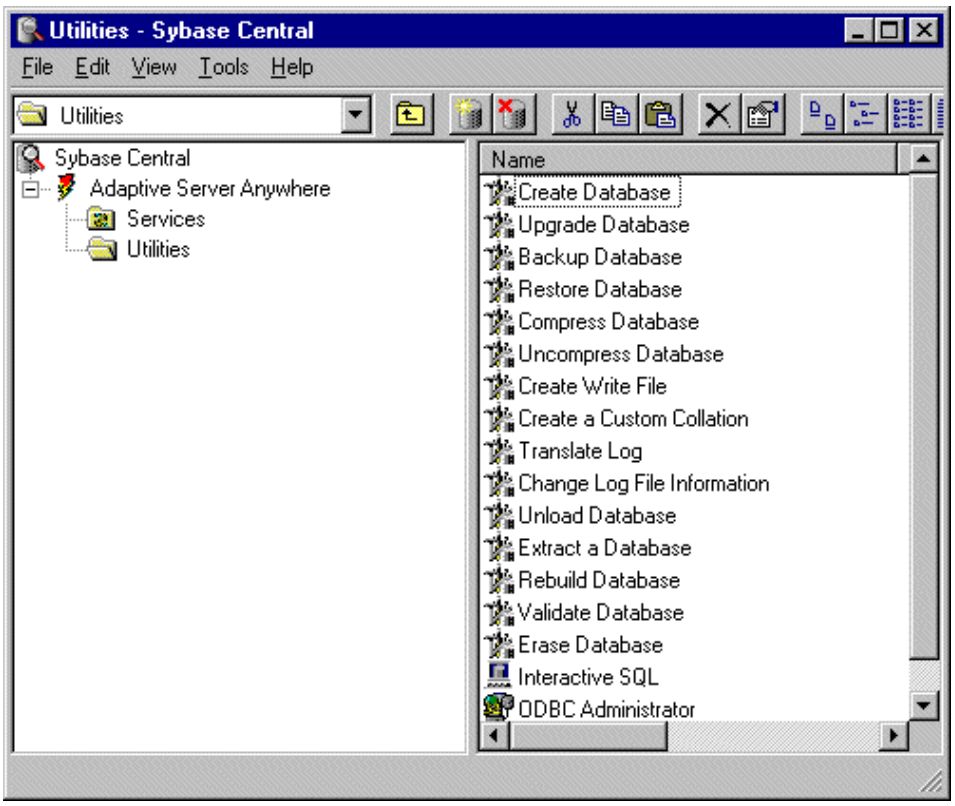


Figure 2. The Sybase Central database administrator

As you can see from Figure 2, Sybase Central gives you an easier way to perform common database administration tasks. If, for instance, you choose the Create Database utility from the right-hand pane, you'll get a Create Database wizard. Click here for a series of images showing the wizard screens.

The Sybase Central Create Database wizard presents a lot of options! But all of these options are just another way of creating a single SQL statement. Listing 1 is the full syntax for SQL Anywhere's CREATE DATABASE command:

Listing 1. The CREATE DATABASE command in SQL Anywhere v.6

```
CREATE   DATABASE   db-file-name
... [
...     [ [ TRANSACTION  ] LOG  OFF  |
                [ TRANSACTION  ] LOG  ON  [ log-file-name  ]
                [ MIRROR  mirror-file-name  ]
        ]
...     [ CASE  { RESPECT  | IGNORE  } ]
...     [ PAGE SIZE  page-size  ]
...     [ COLLATION  collation-label  ]
...     [ ENCRYPTED  { ON  | OFF  } ]
...     [ BLANK  PADDING  { ON  | OFF  } ]
...     [ ASE  [ COMPATIBLE  ] ]
...     [ JAVA  { ON  | OFF  } ]
...     [ JCONNECT  { ON  | OFF  } ]
]
```

If you refer back to Figures 2-10, you'll see that each wizard step corresponds (more or less) to an option on the CREATE DATABASE statement.

By comparison, the full syntax for MySQL's CREATE DATABASE command is considerably less complicated:

```
CREATE DATABASE [IF NOT EXISTS] db_name
```

Modifying database space

If you locate the actual physical file or files that hold your SQL data, you may be surprised to see that there's no correlation between the SQL data file and the tables your database contains. (One notable exception is MySQL, which by default stores each database in its own directory, and each table as a file in that directory.) Since SQL servers are responding to SQL requests, and do not allow the application to read/write the data directly, the server can store the data on disk in whatever way is most convenient.

If your database is small, there's a good chance your SQL server will store the entire database as a single physical file. Larger databases may be spread over several physical files, which can even be located on other computers. One of the DBA's many responsibilities is to manage these different files (often called dataspaces or tablespaces) to ensure that there's enough space available, and that performance remains good. In most cases a SQL server can be set to grow its data files as needed, but there may be predefined limits on how large a given file can get, or you may need to pre-allocate new dataspaces to accommodate anticipated growth.

Database log files

SQL databases typically let you create log files which record all changes to the database. This can be particularly useful in a development environment, when you want to keep track of everything your application actually does with the data. You may also be able to use the log file to re-apply changes to a database which you've had to restore from a backup.

Backing up a database

Just as is the case with TPS files, when backing up a live SQL database you run the risk of creating an inconsistent data copy. It's best to use the backup utility that comes with your database, as this program (hopefully) knows how to get a clean copy of the data. For instance, the backup program may wait for all transactions currently in process to commit before taking a snapshot of the database.

Database replication

I regularly see messages in the SV newsgroups from developers who need to manage one database across multiple physical locations, such as branch offices. This is a common enough requirement in the business world that most SQL servers offer some form of replication.

In a database replication scheme, one SQL server manages the central database, and additional SQL servers manage local copies of that database (or portions thereof). These servers communicate with the central SQL server on a regular basis, exchanging information as necessary.

There are various scenarios for database replication, from simple duplication of data to read-only clients to allowing all servers to fully participate in updating data. As you can imagine, database replication can introduce a whole new set of design problems. Still, it's better to let the server manage this kind of problem than to embody the necessary logic in the user's application.

Summary

The single most important choice you'll make in going to SQL is your choice of SQL server. Although all SQL servers share a common set of SQL statements, vendors typically add their own proprietary enhancements. You should look at a variety of SQL servers before making your choice.

Once you've decided on, obtained, and installed a SQL server, your next task is to create at least one database. You can do this with a simple SQL statement, and you'll probably also have the option of using a utility to create the database. At least you're ready to start creating tables to hold your data, and I'll cover that topic in the next installment.

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

Clarion MAGAZINE

Search
Go

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

The Cranky Programmer: To Patch or Not to Patch, That is the Question

by Cranky

Published 2001-03-06

Well, last week it finally arrived.

Yes, sir, the long awaited first batch of fixes for Clarion 5.5 was actually unleashed upon the world, although there seems to be a bit of confusion on SoftVelocity's part as to exactly what the patch is called. The two most popular names I've seen so far are 5.501 and 5.5a.

When it comes to downloading, though, SV has chosen to name the patch Clarion 5.5a Service Release 1. What a mouthful. Does that mean there will be a C5.5a SR-2? Or maybe a C5.5a SR-1a? And if it is 5.5a, why do we download a file called C5501ee.zip (or C5501pe.zip, as the case may be)?

C'mon, guys – let's pick *one* name and stick to it.

For my cranky old self, I prefer just plain old 5.5a, and indeed, it does seem to be the winner (that's what the Help|About window now says in the IDE). So what's the big hoo-ha about? Let's take a snapshot of the currently evolving situation.

Moving on up

In a perfect world, a patch would simply fix every known bug while having no other impact on your programs.

(Yeah, right - pardon me while I stop laughing hysterically...)

Programming as we do in an imperfect world, it is more often than not an exercise of the law of unintended consequences. That one teensy-weensy change over there that just couldn't *possibly* affect anything else inevitably ends up reverberating through your program like a cannon shot in a cathedral.

And so it goes with this patch. Many bugs were fixed, but the fixing itself flushed out a new set of nasties. The interesting thing

about the newest set, though, is that quite a few of them are a result of fixing longstanding problems in the Clarion compiler itself. Holes where ambiguous code could leak through in the past have been plugged, with the consequence that formerly valid (but syntactically bad) code will now throw a compiler error. This might be painful in the short term, perhaps, but I believe it is definitely a plus for the long haul.

I mean, if you can't trust the compiler, who the heck can you trust?

Let me count the ways

First, a word of warning: The patch is brand new, and the items listed here are culled from early reports on the newsgroups. There may be workarounds that I don't know of yet, or other items that I haven't seen. Stay tuned for more news.

Also, let me state again right here that this patch *does* fix a lot of bugs, including ones that have been hanging for years. From what I can see, SoftVelocity deserves kudos for really starting to clean up the code base.

Then again, there are also a few items that fall under the "Aaaargh! Why did the heck they do *that*?" category...

Color me gone, gone, gone

Depending on your applications, this one may or may not be a showstopper, i.e., a reason to not upgrade yet. If you have a `SHEET` control with the `WIZARD` attribute set (i.e., no visible tabs and you use Next/Back to navigate) *and* you have a color set for the sheet, window or tabs, many of the controls on the sheet will become invisible.

Now, while becoming invisible may be cool for a character in a TV show, it is not generally considered to be good programming practice to have your helpful wizard controls disappear.

Invisible ink

Printing with some dot matrix printers results in blank pages and previews. Wow, just like the invisible ink I used to get at the novelty shop.

There's your way, my way and the compiler's way: Part 1

The next item is related to the tightening up of the compiler, and was reported by Gus M. Creces (author of the Handy Tools templates). It seems that in C5 and C5.5 Gold, using the `WHO()` statement on a queue would return the field name without the prefix, e.g., `MyField`.

In C5.5a, `WHO()` on a queue now returns the field name *and* the prefix, as in `PRE:MyField`.

This change broke some of Gus's templates, so Gus asked the reigning King of Clarion, Alexey Solovjev, if it was a regression or a necessary change. Alexey sent back a scary little program that proved that the C5 and 5.5 Gold compilers could get confused and assign values incorrectly due to the missing prefix (it was guessing, and guessing wrong). So, this one was necessary and can only serve to eliminate possible bugs that would be *very* difficult to track down.

Thanks for sharing, Gus!

There's your way, my way and the compiler's way: Part 2

The following group declaration within an ABC compliant class INC file such works in 5.5:

```
NewGroup Group(Other:Group).
```

The above declaration causes an error in 5.5a, and to fix it you have to exchange the period for an END on the following line:

```
NewGroup Group(Other:Group)
         End
```

Now, the impact of this could be minimal if you're pretty much using pure ABC, since the ABC authors use the second syntax. Or it could be huge if you use third party products that rely the first syntax. It's a gamble, based on the coding style of the third party authors.

If you want more information on some of the above items and how they might affect your own applications, trot on over to the CapeSoft web site where Bruce Johnson has elaborated on these items, their impact on existing C5.5 code, and also their effect on his popular CapeSoft product line. Nice going, Bruce!

There's your way, my way and the compiler's way: Part 3

This one is a flat out bug. It seems there are a few situations where the compiler will now accept invalid spellings, and then the program will GPF when run. The example I've seen is `AND` misspelled as `ANF` in a class method. The compiler accepted the `ANF`, but the running program choked..

And now, in the "Aaargh!" category

I must confess to being totally mystified about one change which causes a *whole* lot of pain for very little apparent gain. As of 5.5a,

the Clarion RED (redirection) file has been renamed. Instead of 5.5's CLARION5.RED, it is now either C55EE.RED or C55PE.RED, depending on whether you have the Enterprise (EE) or Professional (PE) edition.

Now, while the name CLARION5.RED was a bit out of date, it *was* working just fine. With the new change, every single third-party installer that modifies (or talks about modifying) the RED file will need to be re-released. Plus there are now potentially three RED files to deal with instead of one, depending on the Clarion 5.5 version in use. OUCH!!

Further, if you missed the note in the 5.5a patch README file which described this change, you can have all sorts of grief as 5.5a installs a fresh RED file and ignores your prior, modified RED file. Image paths, paths to your existing third party folders, all your redirection changes are pretty much silently bypassed. Double OUCH!!

Can you say "101 instant compiler errors"?

The simplest fix you can do is to install the patch and then copy/rename your old CLARION5.RED file over the new EE or PE RED file. And if you use local RED files (as I do – grrrr) don't forget to rename all those custom CLARION5.RED files too.

The real potential for problems, though, will come a bit farther down the road. Eventually there will be a mix of 5.5a and 5/5.5 installs floating around. You will have to be ever vigilant as to whether a particular install will update the correct version of the RED file. What happens if you install a third party product under 5.5 Gold, and that product is expecting 5.5a and can't find the new RED file? Or you have 5.5a and the product modifies CLARION5.RED?

And all I can keep asking is... why? It's not like EE and PE have radically different needs when it comes to redirection or, for that matter, 5.5 and 5.5a. A BCL (Big Cranky Lemon) to SV for this one.

Stop the presses!!

Late breaking news as of Monday, March 5, posted by Bob Zaunere of SoftVelocity:

> *Just wanted to keep you all informed on the status of reported problems in 5.5a:*
>
> - *SHEET control with WIZARD and COLOR attributes: fixed*
> - *Incomplete update for printing using TTY driver: fixed.*
> - *Possible code generation even when syntax errors were found (Clarion compiler): fixed.*

- *Incorrect processing of GROUP fields by the INC reader*

There are a few other problems being looked at, including the "Invalid record declaration" error in the Database Browser when a file definition is changed.

We also have a few reports that we're looking to get further information on -- to see if they should be included in the update. We'll release the update by end of the week, or sooner.

Well, gee whiz, that covers the first four major items on the above list. With service like this, how can I be cranky with them? Then again, they *did* change the RED file names...

Should I stay or should I go?

The toughest question, of course, is when to take the plunge and install any patch or upgrade. And, as usual, the only possible answer is a definitive "It depends."

It's like the old fable about the tortoise and the hare. There are always some anxious rabbit-types who just *can't* wait to get an upgrade or patch. They post messages along the lines of "Is it ready? Huh? Is it? Is it? Can I have it now, huh, can I?" and generally behave like the first person to download and install the patch will win the next annual "Xtreme Programmer" trophy.

On the other hand are the tortoises. For them, a new release or patch is a grand opportunity to sit back and watch the rabbits end up in the stew.

Speaking of boiling pots, one rabbit (who shall remain nameless) installed the patch, recompiled a program and immediately put it into production without even running the program.

Silly wabbit.

According to the rules of the Cranky Programming Gods, such unrestrained enthusiasm cannot possibly go unpunished. Thus, as this developer told the story, when his staff ran the new version the majority of the data entry forms, etc., were blank (as in "Uh, boss... I think *maybe* you should look at this!"). Yup, he discovered the cute little problem with colors on wizarded sheets. Fortunately, his was a controlled environment and no real harm was done.

So, how to test a patch and not end up on the wrong end of the soup spoon?

Uncle Cranky's handy rules for reducing nagging patch pain

Here's what I like to do (well, maybe *like* is a bit too strong...):

1. Watch the newsgroups for a day or two for an army of singed rabbits yelling about getting burned by fire-breathing, code-stomping Bugzillas. If there are, stop right here and pat yourself on the back for being so smart and patient.
2. Back up your existing Clarion folder by simply making a copy of it on the same drive. That's right, the whole kit and caboodle. (Gee, did you know that MS Word actually knows how to spell "caboodle'?) Name the copy to something like `C55(gold)`.
3. Backup the source folders for the program you want to test with and append something like `(gold)` to the name just as you did with your original C55 folder.
4. Apply the patch to the original version.
5. Re-compile *all* DLLs associated with the test program, using a nice batch compiler like Steve Parker's Go To Lunch. This is important! You are just begging for trouble if you start testing with a mish-mash of 5.5 and 5.5a compiles.
6. Test, test, test, test everything.
7. Test some more in case you forgot something the first four times.
8. Carefully document any new undesired "features" (more on this in the next Cranky Programmer).

If you are happy with the patch, smile, say "Thank you, SoftVelocity!" and resume working. If you aren't, just rename the patched C55 and program folders to something like `C55a (aaargh)`, rename the saved copies back to the original names and voila – you are right back where you were before you started. (You *did* follow steps 2 through 4, right? Good. I thought so.)

Notice that I didn't say anything about making changes to your programs in step 6. The point here is not to start trying out new stuff, it is to validate that none of your existing, already working code got broken during the patch.

At the end of the line

Whether you are a tortoise or a hare, you'll be happy to see that the dedicated souls at SoftVelocity are making genuine progress. And I do believe that this next iteration of 5.5 will turn out to be the most stable version yet, once these initial patch issues have been resolved.

On another front, it's worth mentioning that the 5.5a patch was created with Friedrich Linder's SetupBuilder product, and I have *never* seen a patch go so smoothly. I haven't seen one message yet on the newsgroups about problems with the patch process itself. Solid, Friedrich, solid!

Yes, all this Programming Goodness is almost enough to make me smile.

Almost...

Are you feeling cranky? Do you growl every time you fire up the ol' Clarion IDE? Got a gripe that just won't go away? Share it with your old pal Cranky – it may not fix the problem, but I guarantee you'll feel better!

Reader Comments

Add a comment