Clarion Magazine

Print On Demand Clarion Books: An Odyssey SO

A number of readers have asked for an article explaining how we brought our popular series of Clarion books to print. Dave Harms gives a behind the scenes tour of the production process, and Clarion Magazine's use of print-on-demand technology.

Posted Friday, December 03, 2004

Hotfix 9030 Now Available

A hotfix release to update Clarion 6.1 Build 9029 to version 9030 is now available for download.

Posted Friday, December 03, 2004

An Introduction to Hand-Coding Reports SO

Dermot Herron continues his series for the beginning Clarion programmer with a look at how to easily manage complex reports by using the report template and a little hand code.

Posted Thursday, December 09, 2004

A Copy/Paste Popup Menu SO

Clarion entry controls may have support for Ctrl-C and Ctrl-V for copy and paste, but what do you do when a client asks for the Copy/Paste popup menu? If you're Nardus Swanevelder, you start writing some code.

Posted Friday, December 10, 2004

C6.1 HotFix 9031

A HotFix release to update Clarion 6.1 Build 9030 to version 9031 is now available for download.

Posted Thursday, December 16, 2004

Podcast Production Notes

A number of Planet Clarion listeners have asked for more information on how we record the podcast using Skype. This page now has a production notes section at the end which answers some of these questions.

Posted Tuesday, December 21, 2004

Calling the Skype API SO

By now you've probably heard of Skype, the free internet telephony service that makes the Planet Clarion podcast possible. But Skype is much more than just a free phone call. David Harms shows how to call the Skype API to manage calls, send instant messages, and much more.

Posted Wednesday, December 22, 2004

Using Arrays in Databases SO

Arrays in databases? You can't in SQL and you shouldn't in TPS! But there are times when arrays really do make sense. Dermot Herron shows how to use OVER to get the benefits of arrays with non-arrayed file layouts.

Posted Thursday, December 23, 2004

Looking for more? Check out the **site index**, or **search the back issues**. This site now contains more than 800 articles and a total of over a million words of Clarion-related information.

Your subscription will expire on Tue Sep 20 00:00:00 CDT 2005. Renew now

Planet Clarion for December 15, 2004

Andrew and Dave talk Skype/Clarion integration with special guest Colin

Wynn in the UK, and discuss Clarion/PHP. Will it be the sleeper hit in the SoftVelocity product lineup?

Complete MP3, 00:31:21, 14703 K

A full track listing is available. Paid subscribers can download individual tracks.

For previous podcasts visit the podcast page.

News

Boxsoft Development Named Third-party Vendor of the Week

Product Scope 32 PRO Special

vuFileTools 3.0

Encode Your Email Links

New Softvelocity Drivers Available

CapeSoft Price Increase Looming

File Manager 3 Sybase Support

GPF Reporter

Secwin Online Server!

CPCS v6.11 For C6 9031

cpTracker Sale & News

RPM & AFE for C6 9031

dpQuery 2.02 Full Version

Clarion Developers Challenge Week 14 Winner

IceTips MySetups

dpQuery 2.02 Demo

vuFileTools Version 3 Quick Reference

BST 3.702

SELECT Class

SoftVelocity Products Available Online At ClarionShop

vuAgent And Valutilities Holiday Special

Jaguar Beta 3 Supports Cellphones/PDA Applications

PostgreSQL Demo

EasyListPrint 1.10

Arco Software Named Third-party Vendor of the Week

xTipHotKey Class v2.5

Clarion Developers Challenge Week 12 Winner

Clarion Web Hosting

Search the news archive

Legend

MS	Subscription *or* free membership required
SO	Subscription required

Clarion Magazine

Print On Demand Clarion Books: An Odyssey

by David Harms

Published 2004-12-03

In January of 2004 Clarion Magazine published the Clarion Tips & Techniques book, using print-on-demand (POD) technology. Since then I've received a number of requests for an article explaining how I went about producing and printing that book, and the subsequent Databases and SQL and Programming Objects in Clarion books.

POD is great stuff for anyone who needs to produce books and manuals. It used to be that if you wanted to produce a perfect-bound softcover or hardcover book (that is, one with a glued spine), you pretty much had to resort to a print job of at least 1000 copies to achieve a decent per-copy price. This is no longer the case – with POD you order just the copies you need, as you need them, removing the need to keep inventory. Your per-book costs may well be the same for one book as for a thousand books. Response times can be fast – the printing company I use, Lightning Source, has a two day turnaround, and by the first quarter of 2005 they are promising same-day fulfillment.

Although the information I'll present here is specific to my Clarion books, and some of it only applies if you are producing books that can be sold in bookstores, most of it applies to any book or manual POD publishing venture.

In a nutshell, my book publishing process went like this:

1. Decide on a page layout program
2. Massage existing HTML documents (Clarion Magazine articles) into a print publishable form

3. Design the book
4. Edit/format the content as needed
5. Create PDFs of the cover and content
6. Upload the PDFs to the printing/fulfillment company
7. Modify the ClarionMag store to take book orders
8. Convert the completed book orders into a format suitable for the printing/fulfillment company's EDI interface
9. Send shipping confirmation emails when shipping confirmation is received from the printing company

Becoming a book publisher has certainly been a learning experience, although I'm not completely new to the industry. Back in the late 1980s I briefly worked for a small Canadian book publisher, where my responsibilities ranged from editing to page layout to requesting ISBN numbers and CIP data. And I've had three books published by mainstream publishers, so I've seen the writer's perspective as well. All of that experience proved useful, although I still feel I have a lot to learn, especially about book design.

It took about five months to get the first book in print; of course, during this time I was also conducting the regular business of publishing Clarion Magazine. Once I had all my ducks in a row the second and third books took far fewer hours.

Choosing a page layout program

I began by investigating my desktop publishing (DTP) software options. My short list included Quark Express, Corel Ventura, Adobe PageMaker (which was once truly awful for books, but has improved), Adobe InDesign (positioned as the replacement for PageMaker), and Adobe FrameMaker.

I had worked with PageMaker and Ventura years ago, and much preferred Ventura for book-length documents. PageMaker is also a dead-end product now, and PM users are encouraged to upgrade to Adobe InDesign. I evaluated Quark Express, but like PageMaker it seemed better suited to fancy layout than the needs of long, technical documents. In fact, a modest amount of research led me to the conclusion that the best choice for my kind of book publishing is Adobe FrameMaker, an old workhorse which has capabilities InDesign lacks.

But I still had a problem. All ClarionMag articles are, fundamentally, HTML documents. They may start out as Word or RTF documents, but receive final final editing and formatting as HTML documents. So I definitely needed an HTML to DTP migration path. Only FM 7 has no way of directly importing HTML documents (FM 7.1 has since been released, but I haven’t yet upgraded and I don’t know if it has an HTML import function). I tried a couple of HTML to FM add-on products, but they didn’t give me the level of control I needed over the import process.

Happily, FrameMaker supports an XML-like format called MIF, for Maker Interchange Format. All I had to do was write some code to convert my HTML files to MIF files, import those into FrameMaker, and I was off to the races!

Converting HTML to FrameMaker

Of course, converting my HTML documents to MIF files was a non-trivial task. I wanted to massage the HTML a fair bit; this included tasks like enforcing a consistent style for inline code (in the early years I used HTML <font> tags; now I use the <code> tag and a style sheet), creating standard image captions, automatically inserting and positioning images, and so forth. The MIF file specification is extensive, and it did take me a while to figure out all of the intricacies of my particular MIF creation process. Really I only scratched the surface – with MIF files, you can create incredibly complex documents programmatically. If, for instance, you have a database you need to publish in print, FrameMaker + MIF files makes a powerful combination, particularly for things like catalog publishing.

I’m getting a little ahead of myself, however. In order to convert HTML files to MIF files, I had to be able to get at the HTML data, so I started looking around for a good HTML parser, with which I could extract the text and attributes from the original Clarion Magazine articles.

I’d heard a lot of good things about HTML Tidy, but rather than try to find a way of plugging this parser into Clarion, I elected to go with the Java port, JTidy. The server-side code behind Clarion Magazine is also written in Java, and I had a bunch of existing code I could put to work. JTidy, like HTML tidy, validates and cleans up HTML files, so I had the dual benefit of not only parsing, but parsing something that I knew to be acceptable HTML in the first place.

I won’t go into the code behind the conversion from HTML to MIF – it’s fairly complex, and it is quite specific to the way ClarionMag articles are formatted. But in essence I ran the parser on each document, extracted the article text including formatting, images, links, etc, and massaged this into a MIF file. I then ran a batch utility to convert the MIF files to FM files. After that I was ready to start producing books.

Creating a book

There are a lot of little (and big) tasks that go into creating a book, or at least one that’s fit to print and sell in stores (and although at present you can only buy ClarionMag books online, they are fully and completely ready for bookstore sale also). These tasks include:

1. Editing the content (hopefully already done to some sort of standard, in this case!)
2. Deciding on a page layout, including top, button, side, and gutter margins
3. Choosing fonts
4. Choosing heading styles and levels (page headings, section headings, chapter headings, etc)
5. Indexing
6. Page, chapter, and figure numbering (styles, when to restart, etc)
7. Creating a table of contents

8. Getting an ISBN number
9. Getting the Cataloguing in Publication (CIP) data (optional)
10. Assembling the front matter (all that stuff before the first chapter)
11. Designing a cover

I've probably forgotten a few important points. In any case, a lot of the above items involved getting to know FrameMaker's peculiarities. Like Clarion, FrameMaker has an avid if small following, and there are a few add-on products you really don't want to be without. In particular I made heavy use of Silicon Prairie's Index Tools Professional – without this product, indexing the ClarionMag books would have been painful.

One of my goals for the ClarionMag book series is to have them available for sale in bookstores, and that means each book needed its own International Standard Book Number, or ISBN. ISBNs are typically issued by an agency in your country. I obtained my ISBN numbers from the National Library of Canada. I also obtained Cataloguing In Publication (CIP) data from the National Library, which puts the title into government and library catalog systems.

As an aside, after obtaining ISBN numbers and CIP data, I received directives from both the Government of Canada and the Government of Manitoba to provide copies of my books for legal deposit. This is a legal requirement for all books published in Canada. So all authors whose articles appear in these books are now enshrined in at least two government libraries.

The book cover

Technical book covers are, to put it bluntly, a problem. Often the content of the book is so abstract that no graphic image readily comes to mind. O'Reilly, one of the largest computer book publishers, has solved this problem by using monochromatic animal images on all its covers – as good a solution as any, I think. In a somewhat similar vein, I decided to use pictures of puzzles on all ClarionMag book covers.

As with almost every aspect of my book production odyssey, shooting the covers proved to be a bit tricky. All I wanted was a picture of the puzzle itself, with no background, suitable for placement on the book's white cover. I took the puzzle outside in bright daylight, placed it on a large white sheet of paper, and began shooting. When I got the exposure right on the puzzle, the background was too dark. If I overexposed to wash out the background, I lost contrast and detail on the puzzle.

There is a much better way to do this, as I have since learned. What I needed was a specialized sort of light table. Figure 1 shows such a table, which I built from ½" PVC pipe and ¼" acrylic sheet. I purchased the pipe and the acrylic sheet at my local home improvement center, and I bent the acrylic to shape with careful application of heat from a propane torch.

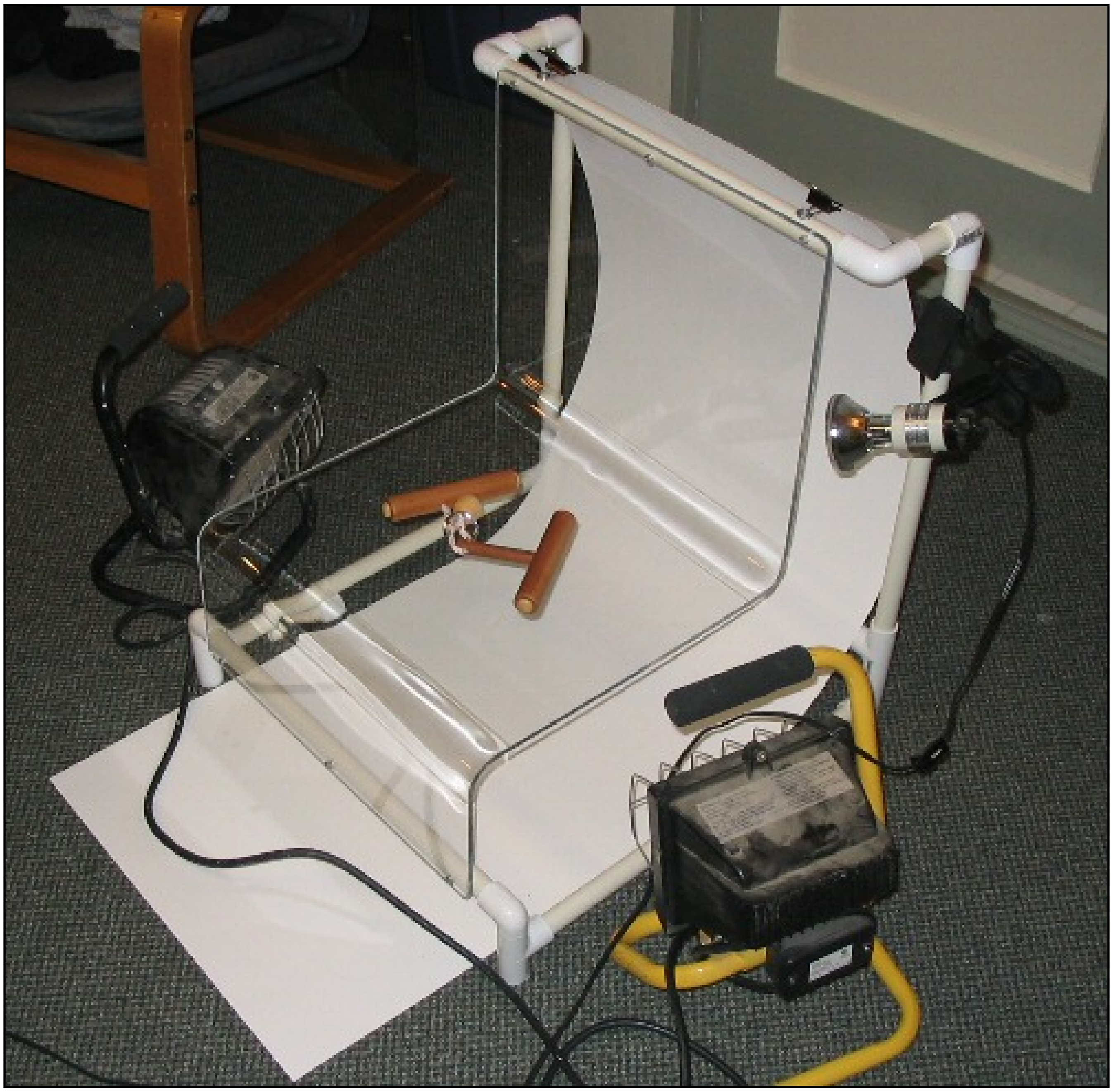


Figure 1. The light table

Listen to Dave's comments on the construction of the light table. MP3, 3255K

The trick to using such a light table is to put most of the illumination on the background (white paper, in this case). I use two 500 watt halogen work lights borrowed from my workshop for background lighting, and a couple of small lamps to light the object itself. There are no shadows, because the object is suspended above the white background. And if the background is bright enough, exposing for the light falling on the object results in a completely (or nearly so) washed out background.

Without the light table, the best I could do is the image in Figure 2, which needed a whole lot of tweaking and retouching before I got an image I could use on the cover.

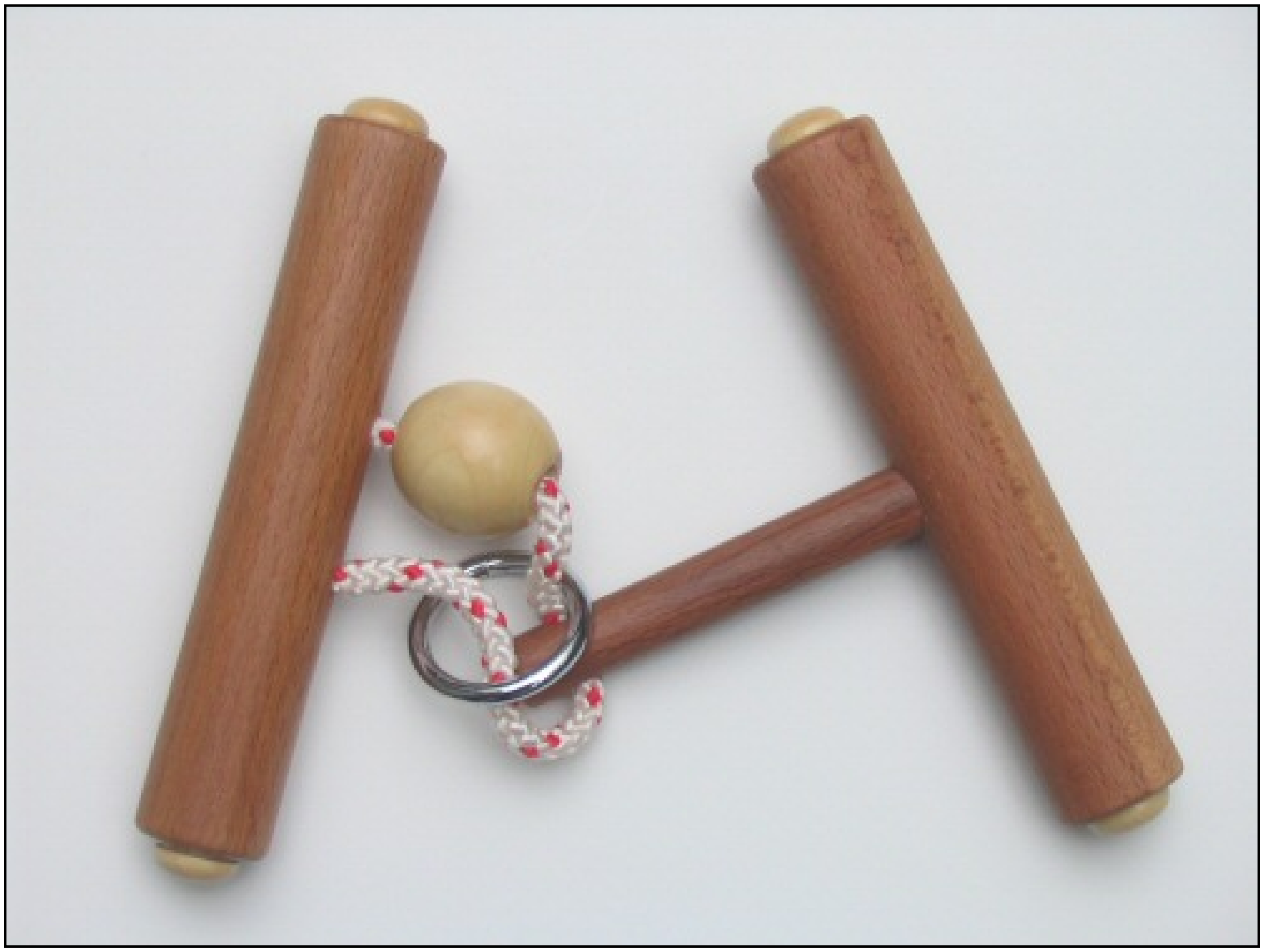


Figure 2. Cover shot on a white background

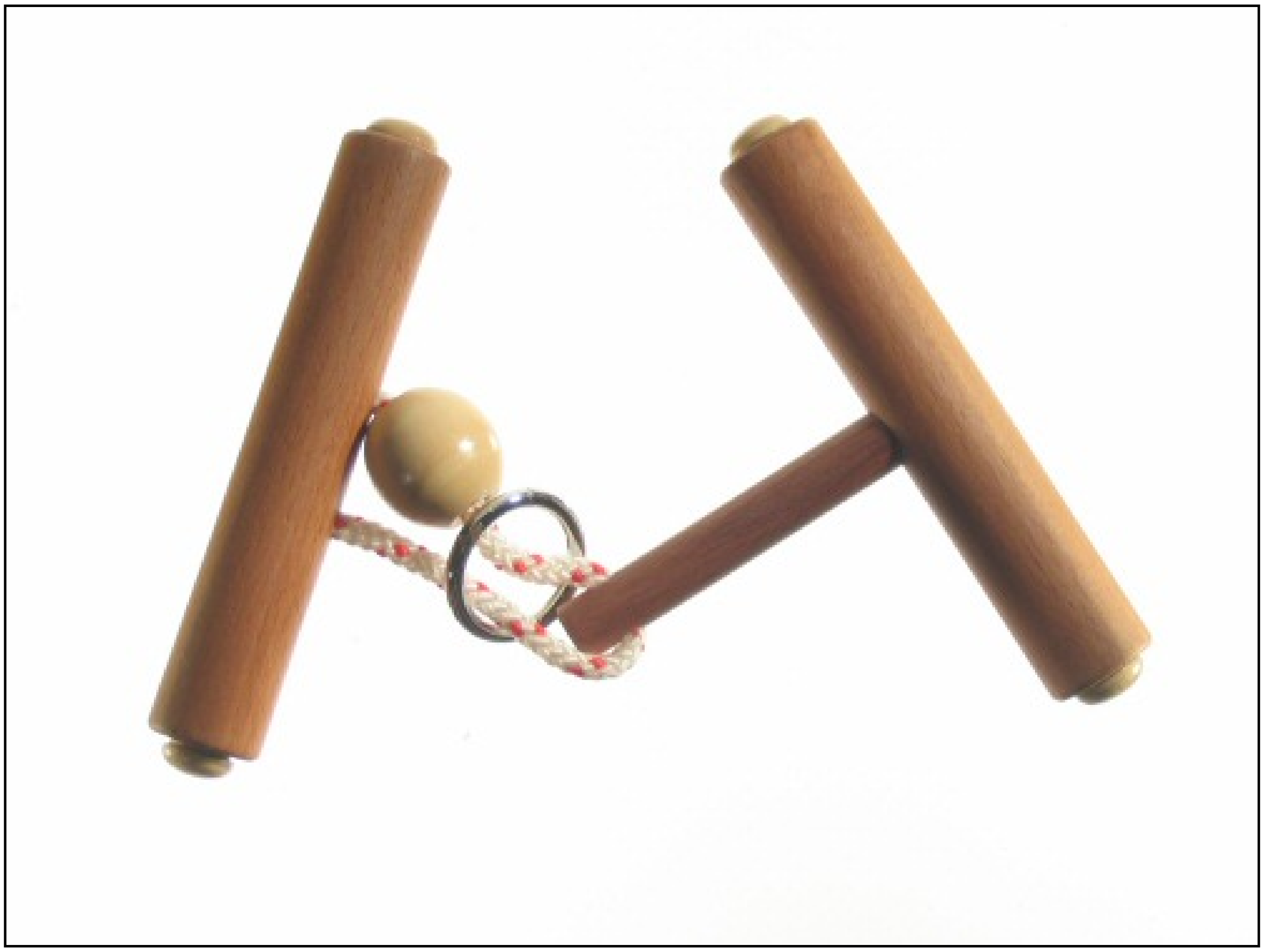


Figure 3. Cover shot taken on the light table.

Figure 3 shows the light table version. Actually the background in that image is not completely white; in HSB terms, it's only 254/254/254 instead of 255/255/255! This is just a quick shot, not something quite ready for a book cover – it needs a bit more care in the lighting, I'd probably also want to at least up the contrast a bit. But as you can see, the light table not only lets you wash out the background, it removes the shadows as well.

Creating the PDFs

Most, if not all, POD companies will accept PDF files. I created two of these, one for the book content, the other for the cover. Lightning Source has an online cover template generator which works well. Templates are available in several formats, and Encapsulated Postscript (EPS) worked well for me, although I had to convert the EPS files to TIFF files as FrameMaker didn’t like this particular EPS variant. I did use one small trick, however. The template included two bar codes, but they weren’t where I wanted them on the back cover, so I first imported the EPS template at 600 dpi and saved the bar codes as TIF files. Then I imported the entire template at 72 dpi which took a whole lot less memory when used as the background for my cover layout (which I also did in FrameMaker).

I printed the PDFs at 600 dpi, and made sure I embedded all fonts. If you don’t embed the fonts, substitutions may be made at the printer and you could end up with a printed document quite unlike the PDF as it displays on your computer.

Fulfillment

I love automation. And the last thing I wanted to have to do with book purchases was manually fill out some form for each book order. What I wanted was a way for customers to buy books on my web site, and for those orders to flow directly to a company that would both print and ship my books.

Lightning Source both prints and ships books, but when I first inquired about the fulfillment system, I was told that I would have to place each book order manually, using their web site. That was unacceptable, so I instead planned to use iFulFill.com, a fulfillment company that would handle my orders in an automated fashion. I anticipated keeping iFulFill.com stocked as needed, so I’d only need to manually place bulk orders instead of individual orders.

Granite Bear's POD Manuals

Granite Bear's Mark Riffey reports that his company has used POD for years for its perfect bound manuals, printing a two volume set (totalling over 700 pages) through Trafford, a Canadian company with offices in North Carolina, Ireland and the UK. "International shipping

But as I got into the Lightning Source documentation, I realized that there was an automated fulfillment process available; I just wasn’t being told about it. I called my rep, who explained that the EDI system was available only to publishers with a certain minimum number of titles and sales; they no longer offered this service to small shops because their tech support people invariably ended up writing code for the publishers, who didn’t have the resources to build their own interface. So I dragged out my credentials, including the books I’d written on software development, and Lightning Source agreed to let me have a crack at it. The code wasn’t difficult to write, the testing went smoothly, and soon I had a system that could take orders from the Clarion Magazine store, upload them to the LS server, and download order and shipping confirmation notices.

As with any new venture, there were a few bumps on the road. When I added the second (Databases & SQL) book, and pre-sold a number of copies, I ran into a problem where customers bought both the shipping book and the not-yet-shipping book. When it came time to ship the second book, all those orders came back with duplicate order number errors, and I had to write a process to move those books to new orders. I had to make significant modifications to the Clarion Magazine store as well – once I began selling product that required physical shipment, I needed to let users confirm their

is a pain, but otherwise Trafford has been good to work with," says Riffey.

"Granite Bear prints PDFs for the manuals right out of Help and Manual, which has been a great bonus. Help and Manual has a template editor, so you can format your margins and related items in their program without affecting other PDFs or help files, and without taking the time to learn another program and reformat your document. The last thing I needed was to have to deal with another piece of software just to format and print manuals. One limitation to be aware of about POD vendors is that their equipment tends to have a pages-per-book limit. Trafford's is about 700, Dave's vendor has a limit in that same shipping address, and keep separate billing and shipping addresses, as well as calculate shipping costs.

range, as I recall. We have Trafford ship us a couple cases at a time simply because we ship the manuals with the other program materials. We dont have to do it this way, we just prefer to do so. Trafford also offers print on demand ordering via their online bookstore. I'm not aware if they publish an API to their ordering system."

Summary

I mentioned that I use Lightning Source for printing and fulfillment. Lightning Source is a division of Ingram, the largest book wholesaler in the US, and offers services that are of particular value to me as a book publisher. There are, however, many other POD companies (see the sidebar), and I have no doubt that some are as well or better suited to printing software manuals.

Print-on-demand technology has made it economical to produce books in small quantities, with fast turnaround. You may not need a complete turnkey solution for selling individual books to customers, as I did. But if you're printing manuals by the thousands to get good pricing, or if you've shied away from perfect-bound books or manuals because of the costs involved, you should definitely investigate print-on-demand.

David Harms is an independent software developer and the editor and publisher of Clarion Magazine. He is also co-author with Ross Santos of Developing Clarion for Windows Applications, published by SAMS (1995), and has written or co-written several Java books. David produces the Planet Clarion podcast, which he co-hosts with Andrew Guidroz II.

Reader Comments

Add a comment

INTERNETSECURE Certified Merchant

clarion mantis

SoftVelocity

Quick Links

HOME

Subscribe/Renew

Clarion Books

FAQ (Frequently Asked Questions)

Log In

Download PDFs

Free Membership

Free Articles

Advertise

Reader Comments

Write For ClarionMag

Privacy

Contact

Clarion Magazine

An Introduction to Hand-Coding Reports

by Dermot Herron

Published 2004-12-09

This is the second in a series of articles that I've written to help the beginning programmer to do something in Clarion in a simple but reliable way. Lots of people have covered these themes in much more detail (see the Getting Started section in the topical index). So I am making suggestions about very *basic* techniques *for beginners*. Do it my way to start with and when you have it right and working, then go on to get more advanced and complicated. Herein lies the huge power of normal Clarion – instead of a ritual (and miles of code), you get a template, and it does the job the same time-tested way every time. But sometimes you *have* to hand-code, and these techniques have served me well.

In the previous article in this series I covered the hand-coding of retrieving records from a file, and talked of *rituals* to establish good and reliable coding practice. This article elaborates on a method of doing reports in Clarion that needs some hand-coding and uses record-retrieval, but makes it very easy to control the break points and control *exactly* what gets printed and when.

I used to think you either used the Report template (which I could never get to work on a real-life report) or you had to code completely by hand. I shied away from all the work and understanding that hand coding meant, but thanks to lots of help from the members of the Johannesburg Clarion User Group (most of whom were also unable to make complicated, reliable reports in C55), I learned that the easiest way to hand-code a report is to use the Report Template! This provides the Preview and all the nice things about the Report Template, including the ability to populate fields, but gives total, easy control over what gets printed where.

The basic principle is to run the Clarion report on *only* the slowest changing table in the report and then do the "faster" bits by hand. Knowing where to put the code is the trick.

I prefer to use the default settings for "A reporting procedure" when specifying the template type. I want to populate the fields needed by hand, and the wizard tends to put in too much.

In the report's file schematic there must be only one table under Report Procedure, with no relationship specified here for that table. (There can be relations specified in the dictionary, but they are not used at all in the report.) All the other files that are going to be used must be specified in Other Tables, including any related files of the main table if you need them, because you will be hand-coding any relations.

Do not put `BREAK`s or `TOTALLING` in the report – you only complicate your life. I will show you how to do all totaling and `BREAK`ing etc. by hand. Delete the "Page-Form" unless needed (remember that this is a simple report for beginners). The database structure used in the example here is shown in Figure 1.

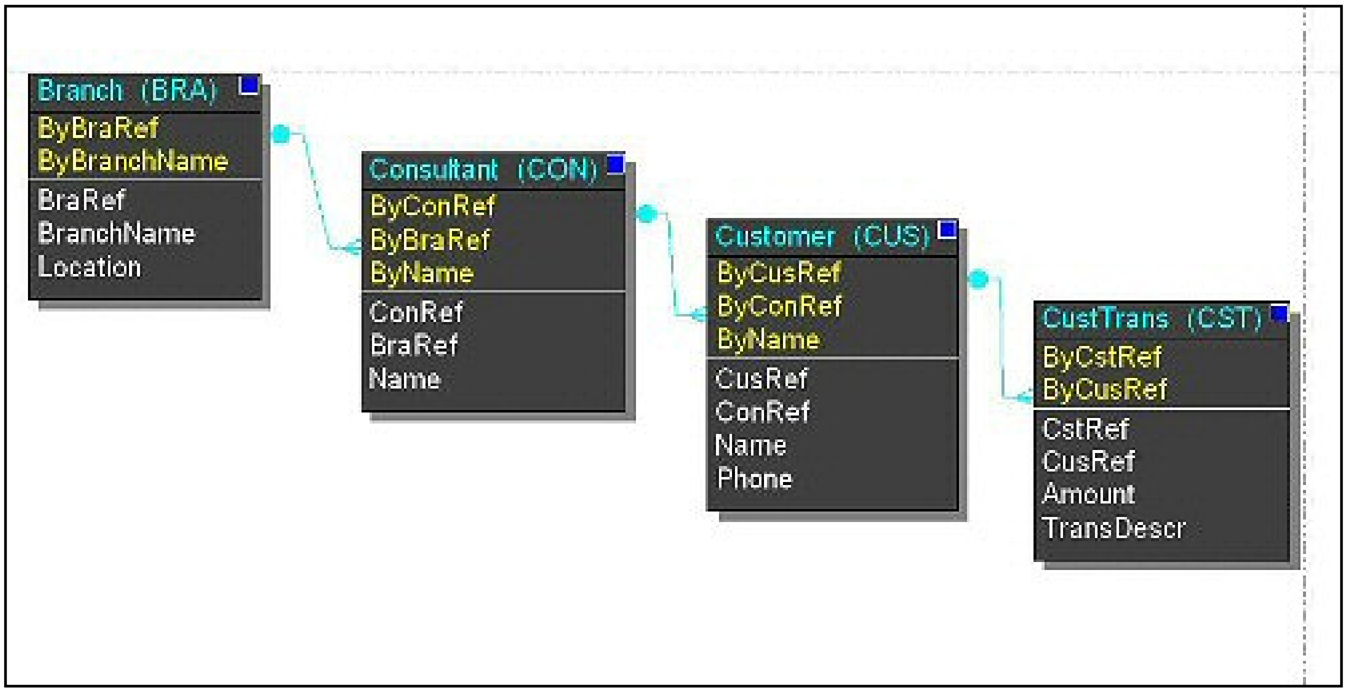


Figure 1. The database structure. (See the full size image)

As Figure 1 shows, a `Branch` has `Consultants` who have `Customers` who have `CustomerTransactions`, each being a one-to-many of the table below it.

The report must print each branch, with the total amount spent by customers showing each consultant, the consultant's total and customers, and how much they each spent. If you know how to do this with `BREAK` etc., stop reading now because you are much cleverer than I am!

Each Clarion cycle in the report calls `ThisReport.TakeRecord` with a new record from the `Branch` file loaded.

The idea is to `OMIT` the `PRINT` commands created by the template and then to do all the printing in hand-coded bits exactly where you want it to be. All the rest of the template processing stays in place.

To find where to put the source-code, create the report and then, using the embed editor, search for `PRINT(` . Then put `OMIT` around the `PRINT`, using the *two* embed points. The `PRINT(RPT:detail)` actually occurs in `ThisReport.TakeRecord` between embeds with priority 5500 and 8000, but it is quicker to search!. This is what it looks like:

```
OMIT(';;;;')
  PRINT(RPT:BranchName)
  PRINT(RPT:ConsultName)
  PRINT(RPT:BranchSummary)
  PRINT(RPT:ConsultantSummary)
  PRINT(RPT:CustomerName)
  PRINT(RPT:CustomerTrans)
  PRINT(RPT:CustomerSummary)
  PRINT(RPT:FinalSummary)
  ! [Priority 8000]
    ! ;;;;
```

Then all the hand-code appears *above* the OMIT statement.

For every different line I wanted printed I made a DETAIL band and named it, *both* in the Label and the USE. There are generally three bands for each database – the name line, the details (possibly several lines) and the total line.

Think of the details as representing nested loops – it looks complicated, but each level is very similar to the next level down. I am processing the "bigger" group, and then inside that the next level down, and so on. So I start at the branch level and print all the consultants. When printing the consultant I print all the consultant's customers. When printing the customers, I print all the customer-transactions. And at the completion of each level, a simple BREAK exits up to do the next one!

The example report is shown in Figure 2.

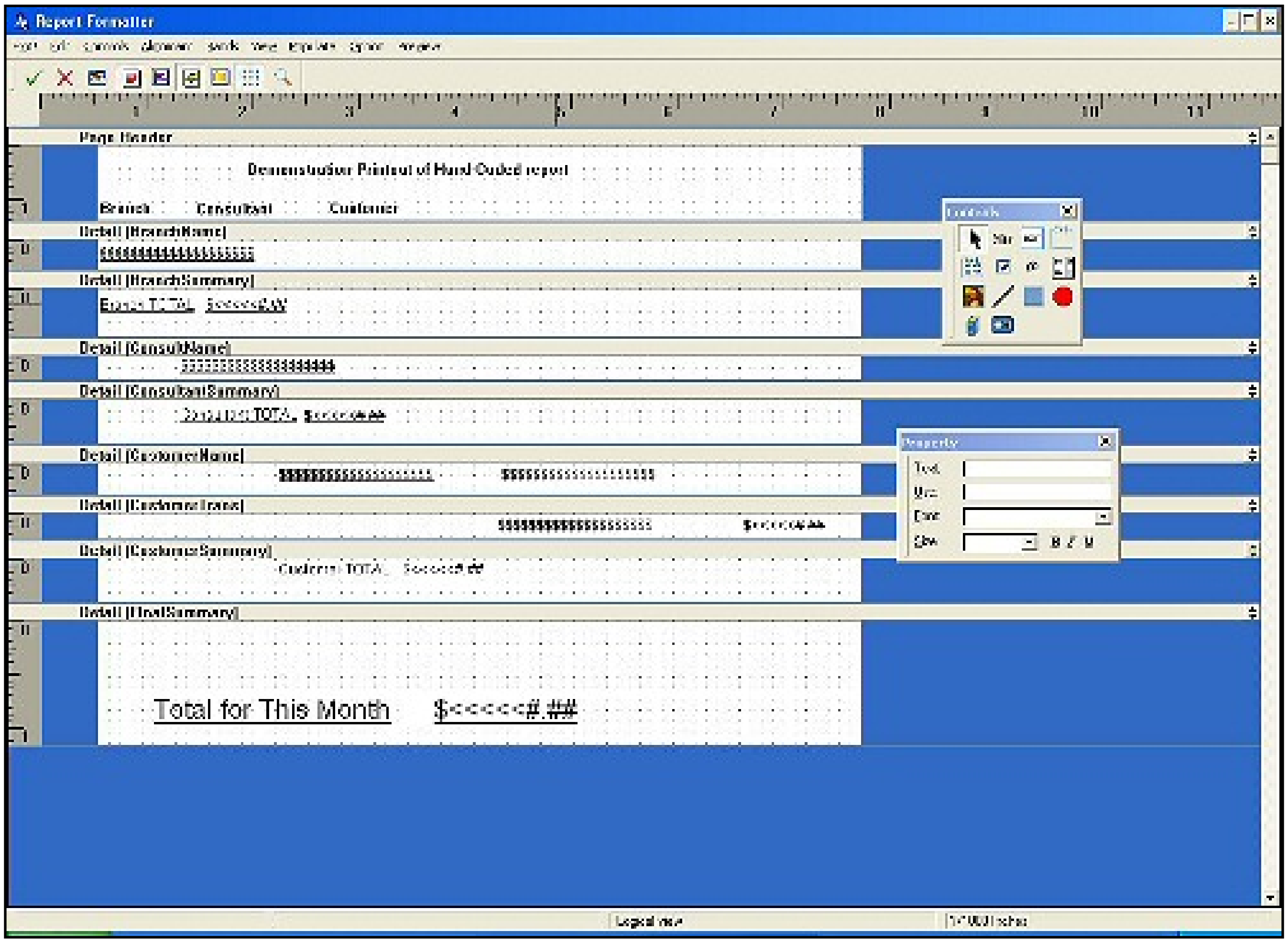


Figure 2. The example report. (See the full size image)

The TakeRecord method

The TakeRecord method is called, by the template, for each new Branch record. Here is the Clarion-generated code:

```
ThisReport.TakeRecord PROCEDURE
ReturnValue          BYTE,AUTO
! Start of "Process Method Data Section"
! [Priority 3500]
SkipDetails BYTE
! [Priority 8500]
! End of "Process Method Data Section"
  CODE
  ! Start of "Process Method Executable Code Section"
  ! [Priority 500]
  ! Parent Call
  ReturnValue = PARENT.TakeRecord()
  ! [Priority 5500]
  ! My code starts here
```

First print the name of the branch:

```
  PRINT(RPT:BranchName)
```

Then zero this branch's total:

```
  lBranchTotal = 0
```

Now loop through all the CONSULTANTS for this BRANCH. Keep in mind that SET(key, key) *starts* at the match specified and *continues* in sequence of CON:BraRef. Remember that the first parameter in this use of SET determines the sequence, and the second parameter determines the starting point:

```
  CON:BraRef = BRA:BraRef
  SET(CON:ByBraRef, CON:ByBraRef)
  LOOP
    CASE Access:Consultant.NEXT()
    OF Level:Benign
```

If the code has gone on to the next branch's consultants, it is finished the current branch so print the accumulated data as a Summary. This also happens if the code arrives at the end of the file (Level:Notify) below

```
      IF CON:BraRef <> BRA:BraRef
        PRINT(RPT:BranchSummary)
```

Break out of the loop, going on to the next BRANCH (which is done by the report template)

```
        BREAK
      END
    OF Level:Notify
      PRINT(RPT:BranchSummary)
      BREAK
    OF LEVEL:Fatal
      STOP('Access:Consultant.NEXT() failed')
```

Don't leave this as *just* a STOP - the user *always* pushes IGNORE, and this often leads to peculiar results.

```
      CYCLE  ! or however else you want to handle a FATAL error !
    END
```

Now it's time to look at the CUSTOMERS of the CONSULTANT record currently in memory; all the not-valid-consultants have been eliminated by now. This loop is very similar to the CONSULTANT section above.

First, print the name of consultant:

```
    PRINT(RPT:ConsultName)
```

Zero this consultant's total:

```
    lConsTotal = 0
```

Retrieve the customers of this consultant only

```
    CUS:ConRef = CON:ConRef
    SET(CUS:ByConRef, CUS:ByConRef)
    LOOP
      CASE Access:Customer.NEXT()
      OF Level:Benign
```

Note the "<>" - if (like me) you copy from just before the SET, it is easy to leave it as "=", and this is a surprisingly difficult bug to find!

```
        IF CUS:ConRef <> CON:ConRef
```

Now print the consultant summary. This is done here and in Level:Notify because there are two possible conditions for the ending – namely a new customer and the end-of-file.

```
          PRINT(RPT:ConsultantSummary)
          BREAK ! this loop to go to the next consultant
        END
      OF Level:Notify
```

```
        PRINT(RPT:ConsultantSummary)
        BREAK ! this loop to go to the next consultant
      OF LEVEL:Fatal
        STOP('Access:Customer.NEXT() failed')
        BREAK
      END
```

Now process all the transcactions for this customer – first print customer name:

```
      PRINT(RPT:CustomerName)
```

Zero this customer total:

```
      lCustTotal = 0
```

Find all the transactions for this customer – again similar to the above code:

```
      CST:CusRef = CUS:CusRef
      SET(CST:ByCusRef, CST:ByCusRef)
      LOOP
        CASE Access:CustTrans.NEXT()
        OF Level:Benign
          IF CST:CusRef <> CUS:CusRef
            PRINT(RPT:CustomerSummary)
            BREAK
          END
        OF Level:Notify
          PRINT(RPT:CustomerSummary)
          BREAK
        OF LEVEL:Fatal
          STOP('Access:CustTrans.NEXT() failed')
          BREAK
        END
```

Keep a running total of the customer transactions – note this was zeroed for each new customer above:

```
        lCustTotal += CST:Amount
        PRINT(RPT:CustomerTrans)
      END
```

Keep a running total for the consultant, also zeroed above:

```
      lConsTotal += lCustTotal
    END        ! of Customer loop
```

And a running total for the Branch:

```
    lBranchTotal += lConsTotal
  END    ! of Consultant loop
```

And lastly an overall total for the whole company:

```
  lOverallTotal += lBranchTotal
```

Remember to prevent the report from doing *any* printing - do *all* the printing by hand above (there is a small danger here that future versions of Clarion may introduce extra embed points, so messing up the coverage of the OMIT, but this should be easily fixed when it occurs):

```
      OMIT(';;;;')        ! this is mine
  PRINT(CRP:NameDetail)    ! these PRINTs are generated
  PRINT(CRP:TransDetail)
  PRINT(CRP:EndDetail)
  PRINT(CRP:TransTotal)
  ! [Priority 8000]
     ! ;;;;            ! this is mine
  ! End of "Process Method Executable Code Section"
  RETURN ReturnValue
```

When all the Branch records have been processed, the following code prints the final calculations at the end of the report:

```
ThisWindow.Next PROCEDURE

ReturnValue          BYTE,AUTO
! Start of "WindowManager Method Data Section"
! [Priority 5000]
! End of "WindowManager Method Data Section"
  CODE
  ! Start of "WindowManager Method Executable Code Section"
  ! [Priority 1300]
  ! Parent Call
  ReturnValue = PARENT.Next()
  ! [Priority 6300]
  ! my code follows
```

This is after the call to PARENT - If the processing of the BRANCH database is complete, `ReturnValue` is set to `Level:Notify`, so I can now finish off the report

```
  IF ReturnValue = Level:Notify
     PRINT(RPT:FinalSummary)
  END
  ! end of my code

  ! End of "WindowManager Method Executable Code Section"
  RETURN ReturnValue
```

What I like about this technique is that I *know* at all times *exactly* what the report is doing. I can do whatever calculations I need and be sure of the results (within my programming and debugging competence!) In a recent report, I allowed the program to go completely through the table and I put everything I wanted to print into a queue. Only when the data was finished did I print out the entire queue in the `ThisWindow.Next` procedure. (I did this because I needed to print in a different sort order from that in the table, and it was too complex to make an index on the file. Probably a VIEW would make the whole thing simpler.) While this technique is *only* for those reasonably comfortable with hand-coding, nothing I've presented here is very complex. .

Download the source

Dermot Herron grew up and went to school in Rhodesia, and earned an electrical-engineering degree at Capetown University. In the 1970s he worked for the Rhodesian Post Office, which was then also the telephone company. He was given the use of an HP9100, which was the very first programmable desktop computer, to help with the design of party-line telephones. He totally fell in love with computers then and there. Leaving the Post Office, he traveled for four years, *worked in Canada (as a cabinet maker) and in England (as a microprocessor engineer) and then immigrated to New Zealand. But Dermot learned that once Africa gets into your blood you are doomed! Now he runs his own company in Johannesburg, writing software to send messages to mobile phones. He lives on a 19-acre plot with a river, and spends his off-time fixing things.*

Reader Comments

Add a comment

Very useful article. The blurb metions that this is a...
The link to the first article is in the second...

Quick Links

HOME

Subscribe/Renew

Clarion Books

FAQ (Frequently Asked Questions)

Log In

Download PDFs

Free Membership

Free Articles

Advertise

Reader Comments

Write For ClarionMag

Privacy

Contact

Clarion Magazine

A Copy/Paste Popup Menu

by Nardus Swanevelder

Published 2004-12-10

Clients! Can't live with them, can't live without them. One of my clients asked me if it is possible to right click on a field and choose copy or paste. My first reaction was to tell him to use Ctrl-C and Ctrl-V, but as always the client is king.

One option is to change all entry fields to TEXT controls. Since TEXT controls are native Windows controls, the standard options are available when you right-click: Undo, Cut, Copy, Paste, Delete and Select All. But TEXT fields have problems too – among other things, they allow multiple lines, and they do not support pictures.

As I looked for an entry field solution, the first thing that jumped to mind was the fact that Clarion has a popup class. I was under the impression that the popup-class only works on browses and list boxes, but I thought that it was a good place to start.

The Popup-Class Overview in the Clarion help file has the following:

> The PopupClass object defines and manages a full featured popup (context) menu. The PopupClass object makes it easy to add fully functional popup menus to your procedures.

So far so good, it looks like it might be able to work on more than just browses and list boxes.

The only other interesting stuff for me in the overview is as follows:

> The PopupClass source code is installed by default to the Clarion \LIBSRC folder. The PopupClass source code and its respective components are contained in:
>
> ABPOPUP.INC PopupClass declarationsABPOPUP.CLW PopupClass method definitionsABPOPUP.TRN PopupClass translation strings

Something that is not in the overview is that there is also an ABPOPUP.TPW file in the Clarion\Template folder

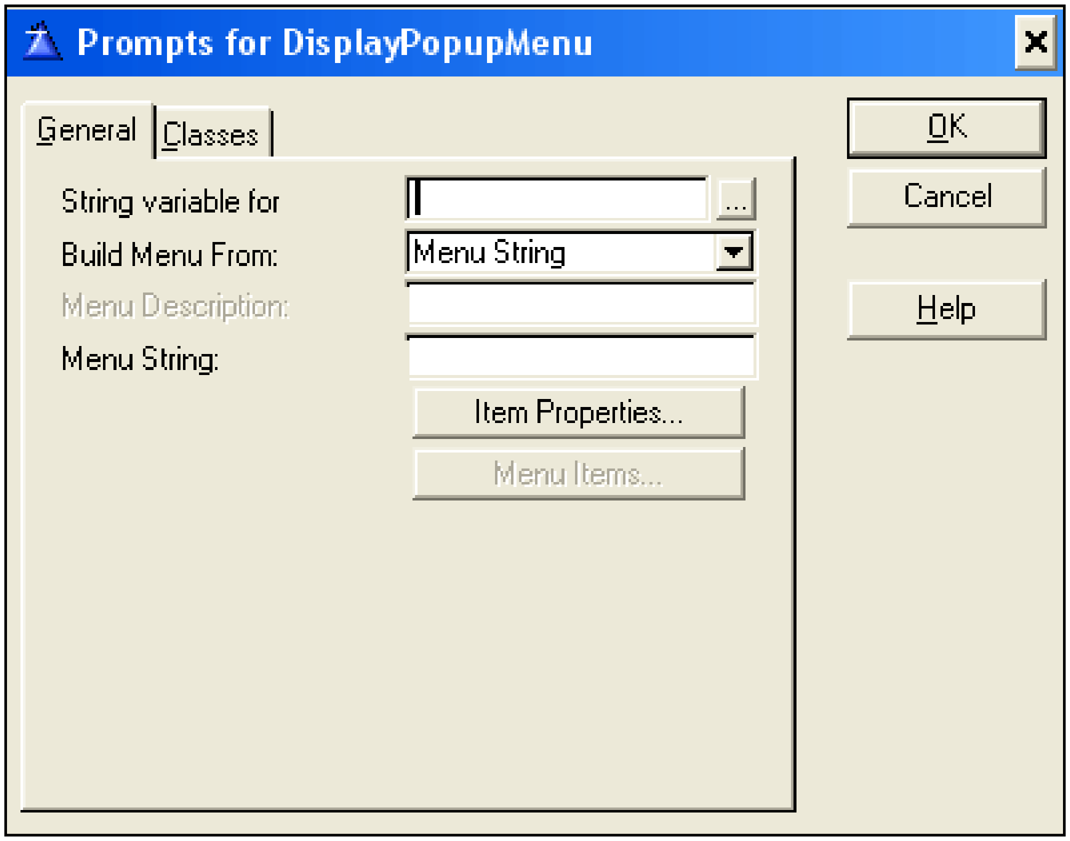
I can’t remember how I found it, but if you look in the Clarion 5.5 Template Guide under Code templates, or you look in the Clarion 6.1 online help under Template Guide, Code Templates, you will find a code template called `DisplayPopupMenu`.

This is what the help has to say about this template:

> The DisplayPopupMenu template generates code to define and display a popup menu, and optionally, act on the end user's selection. You can set the popup menu items to mimic existing buttons on the window so that the associated menu item text matches the button text, is enabled only when the button is enabled, and, when selected, invokes the button action.

Was this what I was looking for? Let’s see how to implement this template.

Open the application that is included in the source download at the end of this article. Open the `UpdateCustomers` Procedure, open the Window editor and go to the First Name field on the form. Go into the embeds for that field, select the event where you want to insert the code template (choose `Accepted` for now) and choose the `Popup` template under Class ABC – Application Builder Class Templates. You will be presented with the screen in Figure 1.



String variable for:
Build Menu From: Menu String
Menu Description:
Menu String:

Figure 1. Prompts for the DisplayPopupMenu code template

I will briefly look at some of this template's functionality as the purpose of this article is not to cover the template in full, but to look at how popups can be added to entry fields.

In testing the functionality I have populated the prompts with the following values:

String variable for:	`CUS:FirstName`
Menu string:	`Copy|Paste`

The menu string supplies the popup class with the options that will be display during the popup. Separate multiple menu options with the pipe character |.

The next step is to tell the popup class what it should do with each menu option. To do this you need to click on the item properties button.

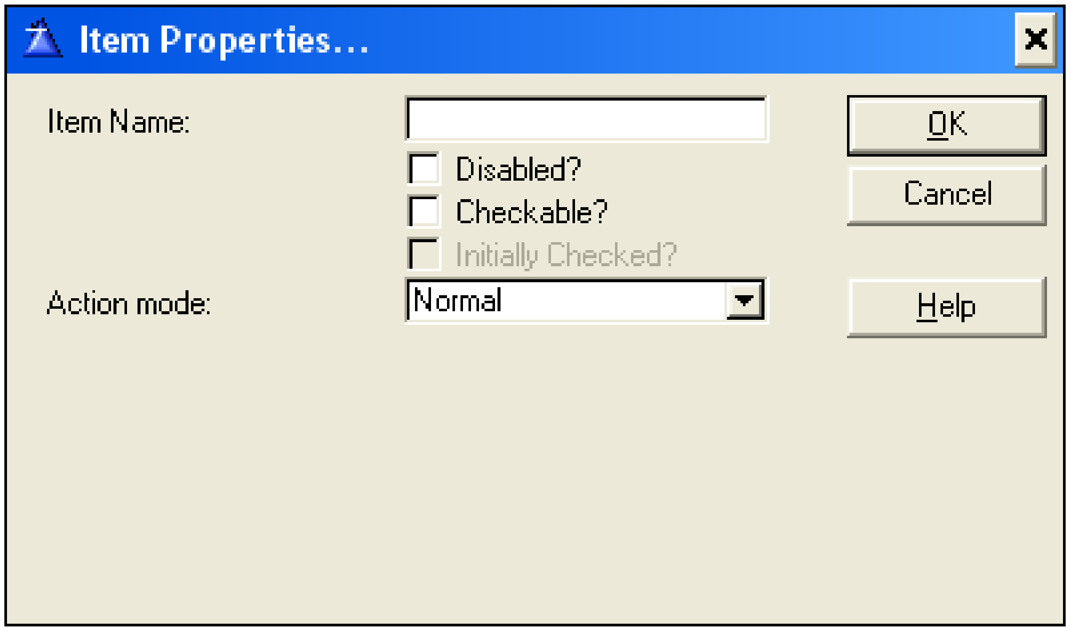


Figure 2. Popup class item properties

I added the following information for the menu items:

Item Name:	Copy
Action Mode:	Normal

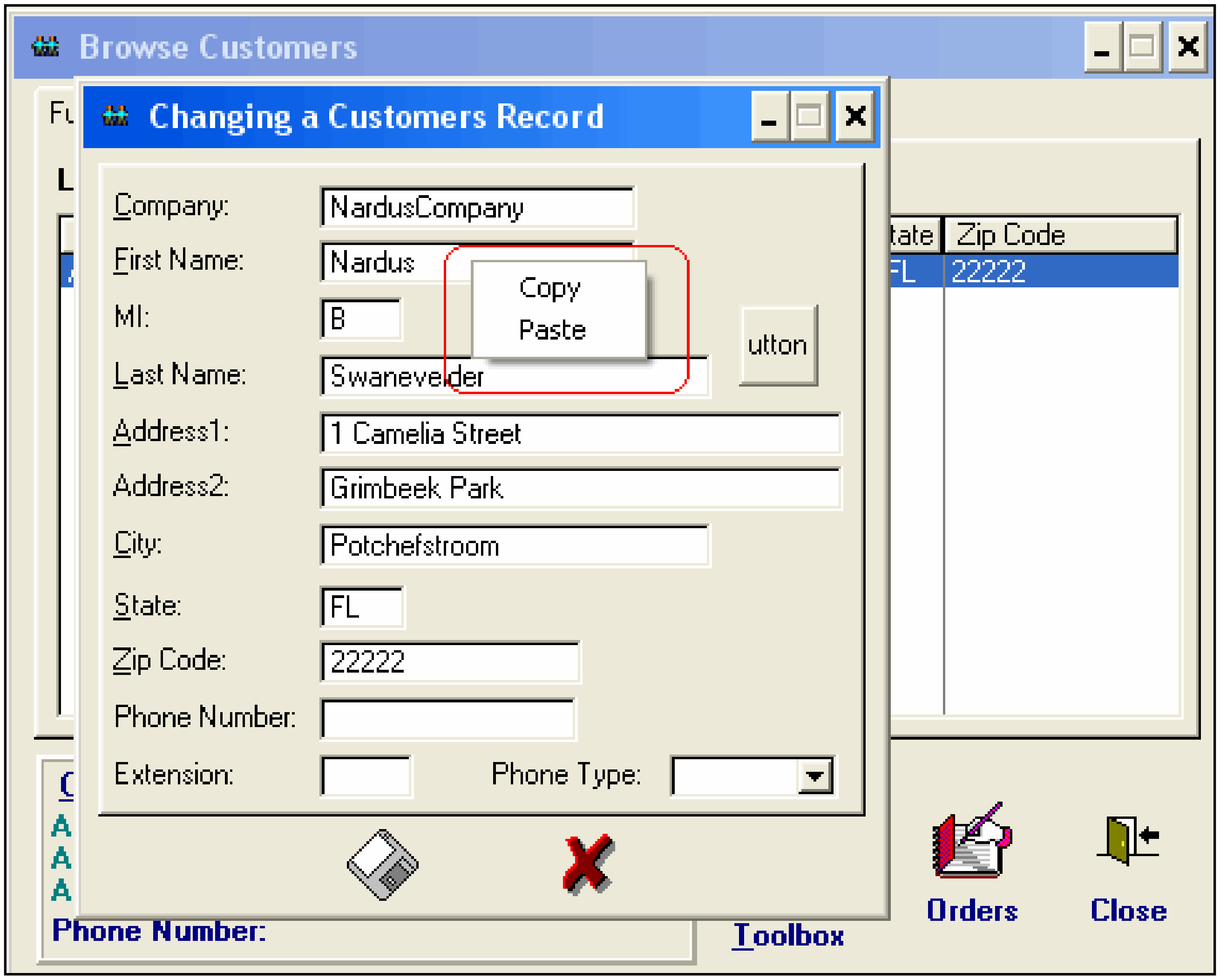
And

Item Name:	Paste
Action Mode:	Normal

An action mode of Normal means that the popup menu will not execute any code, by default, when the user clicks on an option.

Close all the screens, compile and run the application. Insert a new customer.

When you type a first name and hit tab you will see a popup menu appear, with "Copy and Paste" as options.



Company: NardusCompany
First Name: Nardus
MI: B
Last Name: Swanevelder
Address1: 1 Camelia Street
Address2: Grimbeek Park
City: Potchefstroom
State: FL
Zip Code: 22222
Phone Number:
Extension:
Phone Type:

Figure 3. The Copy/Paste popup menu

Note the position where the "Copy, Paste" popup menu is displayed. The PopupMgr.Ask method will display the popup menu at the current mouse cursor, position unless you override the position manually. The PopupMgr.Ask method is populated by the template for you. More on this a little later.

Does this achieve anything? Not really, since the user needs to complete a first name and after that the Copy and Paste menu pops up? The problem is that you want this to pop up when the user right-click with his mouse on the first name, and not when the field is completed.

The first step is to alert the right mouse button. Go back to the

`UpdateCustomer` procedure. Right-click on the First Name field in the window formatter, choose Alert..., click on the Add button, and check the Mouse Right Button option.

Go to the embeds again, delete the `DisplayPopupMenu` template (it is not possible to move it to another embed), and this time you will see two new events – `AlertKey` and `PreAlertKey`. Choose `AlertKey` and add the same code template as before with the same settings.

I suggest you populate the template on `Address1` this time to make it easier to see the difference in functionality.

Close all the screens, compile and run the application. Insert a new customer.

When you click with the mouse's left button on the `Address1` field does anything happen? No. When you type a name and hit tab, does anything happen? Again, no. When you right click does anything happen? Yes, the popup menu appears.

Note that when you click on any of the menu options, the contents of the `Address1` field changes to "copy" or "paste" depending on which one you click on. More on this later.

The reason why nothing useful happens when you choose either of the popup options is that you haven't assigned an action to the menu items yet.

These three actions are available through the template:

- Normal,
- Mimic Button,
- Post Event.

The mimic button is the easiest action to use. Simply populate a button on the window, hide it, and add all the code you need to it's embed points. Instead of clicking the button to make the action happen, you will tell the popup to effectively click the button for you. Lets try it.

Add two buttons, call one ?BtnCopy and the other ?BtnPaste. Hide them and then go to the Accepted embed of BtnCopy. Add the following code:

```
SetClipboard(CUS:Address2)
```

SetClipboard() will take the value of CUS:Address2 and post it onto Window’s clipboard.

Under the Paste button’s Accepted embed add the following code:

```
CUS:Address2 = ClipBoard()
Display(?CUS:Address2)
```

Clipboard() assigns the current value on the Window’s clipboard to CUS:Address2.

Now go back to the popup template and instruct it to mimic the actions of the appropriate button for each popup choice.

Close all the screens, compile and run the application. Insert a new customer.

Right click on the Address2 field and choose any one of the menu options.

Does it work?

No, when you do a copy the result is "copy", and when you do a paste the result is "paste". Why is this happening?

Have a look at the code that is generated by the template for the Cus:Address2 field:

```
OF EVENT:AlertKey
CUS:Address2=PopupMgr10.Ask()
```

What is happening here? `PopupMgr10.Ask` displays the `Popup` menu, and returns the text of what the user has selected and assigns it to `Cus:Address2`. So this is why you are getting a value of "copy" or "paste".

So how can you use the template? One option is to assign a local variable to the "String variable for" prompt and then check the local variable to see what code needs to be executed. Let's try that.

Create a local string variable and call it `LCL:Test`. Change the `CUS:Address2`'s String variable in the template to `LCL:Test`.

Go into the embed point and after the call to `PopupMgr10.Ask()` add the following code:

```
If LCL:Test = 'Copy'
  SetClipBoard(CUS:Address2)
Elsif LCL:Test = 'Paste'
  CUS:Address2 = Clipboard()
  Display(?CUS:Address2)
End
```

Remove the code from the Copy and Paste buttons as it is not needed anymore.

Close all the screens, compile and run the application. Insert a new customer.

Everything should be working fine.

The question that you are asking yourself is "then why do I need a button?" The button, in this case, needs to be there for the template to generate the call to `LCL:Test = PopupMgr10.Ask()`.

This seems silly. Is the Action "Post Event" any better?

As there are no events for Copy and Paste you will have to create user defined events.

The first step is to create equates for two new user defined events. In the local procedure (for now) under the data section you need to add the following equates:

```
Event:CopyField EQUATE (401H)
Event:PasteField EQUATE (402H)
```

Once again I have added the code under a new field to demonstrate the difference in functionality. I added the following code under the `CUS:City` alert embed point:

```
If LCL:Test = 'Paste'
   CUS:City = Clipboard()
   Display(?CUS:City)
ElsIf LCL:Test = 'Copy'
   SetClipBoard(CUS:City)
End
```

Using the Post Event action you end up with Events that are used to generate the call to the `Ask` method, but they are not used for anything else.

Using this approach did not make life any easier for what I wanted to achieve.

To recap - the current solution requires you to create two buttons, hide them and then you don't use them! You also have to add code to each and every field where you require the copy and paste functionality.

This sounds like the work for a template but before I continue I need to define what this new template should do.

The template should specify the field where the functionality is required. It needs to add a popup menu with copy and paste options and it should add the necessary code to the field's embed point.

Where do you start writing this new template? Luckily, SoftVelocity has done most of the work already. Have a look at the code in the `DisplayPopupMenu` template.

Do you want to make changes to the Clarion ABC template file directly? Not a good idea. I created a new template and called it ABPopupNs.tpw. I think it will be the easiest if you open my file and start working through it.

Here is the first part of the new template (with some line breaks added):

```
#CODE(DisplayPopupCopyPasteMenu,↵
    'Displays a popup menu for Copy and Paste'),↵
    HLP('~TPLCodePreparePopup'),MULTI,PROCEDURE
#PREPARE
  #CALL(%ReadABCFiles)
  #CALL(%SetClassDefaults, 'Default',↵
     'PopupMgr'&%ActiveTemplateInstance,%PopupClass)
#ENDPREPARE
#SHEET
  #TAB('&General'),HLP('~TPLCodePreparePopup_General')
    #BOXED('Default Resizer prompts'),AT(0,0),WHERE(%False),HIDE
      #INSERT(%OOPHiddenPrompts)
    #ENDBOXED
    #PROMPT('Variable for Copy && Paste:',FIELD),%MenuIDField
    #DISPLAY()
    #DISPLAY()
    #PROMPT('Declare Local Variable',Check),%LocalVariable,AT(10)
    #PROMPT('Remove formatting at end of paste string',Check)↵
        ,%RemoveFormatting,AT(10)
  #ENDTAB
  #TAB('&Classes'),HLP('~TPLCodePreparePopup_Classes')
    #WITH(%ClassItem,'Default')
      #INSERT(%ClassPrompts)
    #ENDWITH
  #ENDTAB
#ENDSHEET
```

The template declares prompts for some ABC class information, and then it asks for a Variable for Copy & Paste. This is the variable that will receive the paste value and supply the copy value. This is not the local created variable.

The problem I have when I add this code template multiple times to the same procedure is that I end up with the local variable being declared each time. I tried various things but the easiest solution for me was to add the following check box in the template: Declare local variable. If you select the check box the template will declare the variable. So the easy solution is to select this check box only once per procedure.

The last check box asks the following: Remove formatting at end of paste.

What does this do, and why do I need it?

When you copy and paste, for example from MS Excel, Excel copies the formatting characters along with the text, and your user normally does not like that. More on this later.

The next couple of lines (not displayed) are from the original template and can be ignored for now.

The following code declares the local variable in the data section based on the value of `%LocalVariable`.

```
#AT(%DataSection),PRIORITY(4000)
#If(%LocalVariable = 1)
LCL:PopupText        STRING(20)
#ENDIF
#ENDAT
```

I removed a lot of the original code from the next section as this template will always create a menu with the Paste and Copy options.

```
#AT(%WindowManagerMethodCodeSection,'Init','(),BYTE'),PRIORITY(8500)
%PopupObject.Init(INIMgr)
  #IF(%EnableRunTimeTranslator)
%PopupObject.SetTranslator(Translator)
  #ENDIF
%PopupObject.AddMenu('Paste|Copy')
#ENDAT
```

The next section of code adds the call to the `PopupMgr.Ask()` method, checks for the MouseRight key code and then decides if it should do a paste or copy.

Note that this section of code is not encapsulated in a #AT(), #ENDAT section. The template will add this section of code at whichever embed you add the code template.

```
#IF(%MenuIDField)
   ! Might have other alerts on this field as well.
   If KeyCode() = MouseRight
      LCL:PopupText = %PopupObject.Ask()
      Select(?%MenuIDField)
      ! Make sure the correct field is selected
      If Clip(LCL:PopupText) = 'Paste'
        %MenuIDField = Clipboard()
        #IF(%RemoveFormatting = 1)
```

Remove any format characters at the back of the string that is returned from the clipboard.

The formatting characters at the back of the string should normally be <27,15> but I found that it was sometimes padded with other characters as well. The solution that worked the best was to check if the last two characters are alpha characters, and if they are not I remove them.

```
        If Not IsAlpha(Sub(Clip(%MenuIDField),-2,2))
           %MenuIDField = Sub(Clip(%MenuIDField),1,Len(Clip(%MenuIDField))-2)
        End
        #ENDIF
        Display(?%MenuIDField)
      Elsif (LCL:PopupText) = 'Copy'
        SetClipboard(%MenuIDField)
      End
   End
#ENDIF
```

That is al there is to the new template. The last questions that need to be answered are: how do I implement and use this code template in an application?

As stated before you do not want to make changes to ABPopup.tpw, so how do you get Clarion to use the new file?

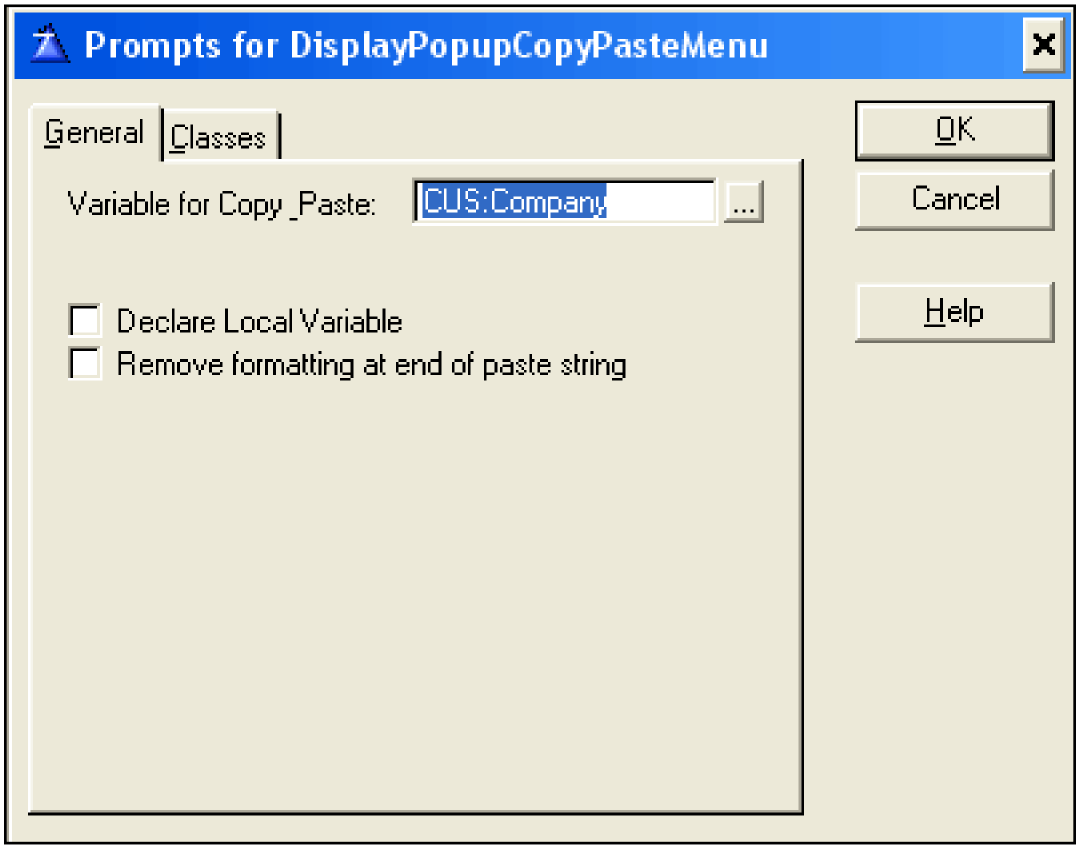
Open the file ABChain.tpl, scroll down and you will find a #INCLUDE statement for ABPopup.tpw. All you need to do is to add the following line underneath that:

```
#INCLUDE('ABPOPUPNS.TPW') #! Popup manager templates
```

Clarion will now read the new template file as part of the normal ABC templates.

Here's how you implement this template on a procedure.

Open your application, go to any update procedure, open the window formatter, select your field, right click and alert the Mouse-Right key. All that remains to be done is to add the new code template (`DisplayPopupCopyPasteMenu – Display a Popup Menu for Copy and Paste`) at the `AlertKey` embed and complete the prompts as per the next screen.


Prompts for DisplayPopupCopyPasteMenu

General | Classes

Variable for Copy_Paste: CUS:Company

☐ Declare Local Variable
☐ Remove formatting at end of paste string

Figure 4. The template prompts

I have added more functionality to the template in the downloadable source, including: Enable Copy, Enable Paste, Force Uppercase on Paste. This enables you to have the Paste option only or the Copy option only per field. I also added Delete and Select All to the popup menu. This demonstrates how easy it is to extend this template to add more functionality.

Summary

This new template gives you the ability to add mouse-right copy and paste functionality to field level on an update form or window. The ability to copy only part of a field or paste part of a string into an existing string is functionality that is not available in this template as of yet.

Download the source

Nardus Swanevelder was born and raised in South Africa. He was a networking engineer for seven years before he moved over to the commercial side of the business. Nardus has developed a Sale Cycle Management system for the Information and Communication Technology industry. He has been programming in Clarion since 1989, and holds B.Com and MBA degrees. In his spare time Nardus lectures Financial Management to B. Com Hons students at North-West University.

Reader Comments

Add a comment

The ABCfree templates have a global extension adding a...
I don't use the ABC Free templates myself. You say it has...
When ndoing the copy: SetClipboard(%MenuIDField) If...
With the addition of the following you can allow dates...
Thanks Paul. I did not think about the picture. There is...

Nardus, Yes, I like the extra ability to do nothing,...
I've updated the source zip with the second version of...
Maarten Veenstra has provided a template for the...
Minor but the template code does not need to handle 1024...

Quick Links

HOME

Subscribe/Renew

Clarion Books

FAQ (Frequently Asked Questions)

Log In

Download PDFs

Free Membership

Free Articles

Advertise

Reader Comments

Write For ClarionMag

Privacy

Contact

Clarion Magazine

Clarion Magazine's Podcasts

What's a Planet Clarion Podcast, you ask? In short, it's an internet radio show for Clarion developers. Basically we're talking about audio programming in MP3 format, which means that all you really need to do is click on one of the links below to download and listen to Planet Clarion. The term *podcasting* was coined to describe the download of these kinds of programs to the Apple iPod, but you don't need an iPod to listen to Planet Clarion. Just click on the links below.

If you want to automatically download the feed instead of coming to this web page and clicking on the links, try some of the RSS software listed at iPodder.org. Point the RSS reader of your choice at our Planet Clarion RSS feed:

http://www.clarionmag.com/planetclarion.rss

Comments? Send us an email

Planet Clarion is hosted by Dave Harms, Clarion Magazine's editor, and Andrew Guidroz II, your favorite Cajun.

Planet Clarion for December 23, 2004

Track	Start	Length	Size	Description
Complete Podcast (Free Access)	00:00:00	00:36:14	16990 K	In this Eve of Christmas Eve edition of the Planet, Dave and Andrew talk at length about the IP driver with Capesoft's Bruce Johnson. Track list now available.

Individual tracks				
Track 1 SO	00:00:00	00:01:39	775 K	Intro
Track 2 SO	00:01:39	00:03:58	1401 K	Bruce Johnson and African languages
Track 3 SO	00:05:38	00:02:59	1050 K	What is the IP driver really for?
Track 4 SO	00:08:37	00:02:22	834 K	Using the IP driver over a WAN
Track 5 SO	00:10:59	00:04:01	1417 K	The IP driver's killer feature
Track 6 SO	00:15:00	00:01:58	695 K	The IP driver and bandwidth issues
Track 7 SO	00:16:59	00:01:02	366 K	The five minute conversion
Track 8 SO	00:18:02	00:03:47	1333 K	Filtering: server side vs client side
Track 9 SO	00:21:48	00:06:09	2164 K	More on bandwidth, saturated LANs, and WANs
Track 10 SO	00:27:58	00:05:25	1910 K	IP driver vs SQL and other technologies, and some tips/tricks
Track 11 SO	00:33:24	00:02:17	974 K	Color theory redux, and closing comments

Planet Clarion for December 15, 2004

Track	Start	Length	Size	Description

Complete Podcast (Free Access)	00:00:00	00:31:21	14703 K	Andrew and Dave talk Skype/Clarion integration with special guest Colin Wynn in the UK, and discuss Clarion/PHP. Will it be the sleeper hit in the SoftVelocity product lineup?
Individual tracks				
Track 1	00:00:00	00:01:27	683 K	Intro
Track 2	00:01:27	00:03:49	1792 K	Colin Wynn's easy way to integrate Skype into a Clarion app
Track 3	00:05:16	00:00:59	466 K	The idylic English countryside
Track 4	00:06:15	00:05:04	2377 K	Removals and storage
Track 5	00:11:20	00:02:28	1159 K	SoftVelocity's new Clarion/PHP product
Track 6	00:13:48	00:09:44	4565 K	Andrew's life with PHP
Track 7	00:23:32	00:04:54	2304 K	If I had a hammer - PHP versus the complicated solution
Track 8	00:28:27	00:01:06	518 K	Clarion/PHP - the sleeper hit?
Track 9	00:29:33	00:01:48	848 K	Closing comments
Track 10	00:00:00	00:06:36	3090 K	Bonus track - Session management and other hazards of web development

Planet Clarion for November 26, 2004

Track	Start	Length	Size	Description

Complete Podcast (Free Access)	00:00:00	40:42:00	19328 K	Planet Clarion #2 is in the can! In this edition Andrew and Dave look at color theory and application skinning, discuss the IP driver, and talk to graphic designer, Clarion developer, and rising photography star Leroy Schulz.
Individual tracks				
Track 1	00:00:00	00:05:08	2408 K	Intro, and a bit of information about how we record the podcast.
Track 2	00:05:08	00:02:53	1360 K	Recording Skype, and Andrew's live lunar eclipse report
Track 3	00:08:01	03:08:00	1472 K	Is that a Clarion application, or a ransom note?
Track 4	00:11:10	00:05:25	2542 K	There's more than one way to skin an application
Track 5	00:16:35	00:06:07	2872 K	Color theory and making your app look cool
Track 6	00:22:42	00:04:53	2294 K	Leroy Schulz, graphic designer to the stars (well, the planets, anyway)
Track 7	00:26:36	00:03:17	1544 K	Leroy's second funkiest domain name
Track 8	00:30:53	00:07:19	3438 K	The IP driver, and what ever happened to Bob Foreman's tie?
Track 9	00:38:13	00:01:24	660 K	Keep those cards and letters coming
Track 10	00:39:38	00:01:04	503 K	Commercial: Never Better!

Planet Clarion for November 9, 2004

Track	Start	Length	Size	Description
Complete Podcast (Free Access)	00:00:00	00:34:00	16813 K	In this first ever Planet Clarion podcast, hosts Dave Harms and Andrew Guidroz II discuss topics ranging from "Why stay with Clarion?" to how many developers are using Clarion, and the impact of Clarion.NET.
Individual tracks				
Track 1	00:00:00	00:01:07	522 K	Intro, with some theme music and a bit of rambling from Andrew and Dave
Track 2	00:01:07	00:04:40	2185 K	Why stay with Clarion? Pre-release anxiety and the wonders of wizarded apps.
Track 3	00:05:45	00:01:22	647 K	Making money by mopping up after other non-Clarion developers
Track 4	00:07:09	00:03:47	1777 K	How many Clarion developers are on the Internet? Long live the lurker.
Track 5	00:10:56	00:02:26	1143 K	DevCon 2004 and Clarion.NET
Track 6	00:12:32	00:01:15	584 K	Rubber boots
Track 7	00:14:37	00:11:14	5269 K	More .NET - it's gonna fark! .NET vs the Win API, exceptions, managed code...
Track 8	00:25:50	00:07:34	3548 K	Clarion does it today. Grok it.
Track 9	00:32:24	00:01:23	650 K	Closing comments
Track 10	00:34:48	00:01:04	502 K	Commercial: "Never Better!" (free access)

Freedom to distribute podcasts

You are free to distribute *public access* podcasts from Clarion Magazine, provided you do not modify those podcasts, and you do not charge any fees for the podcasts. In other words, if you want to put a podcast up on your server, feel free.

You may not, however, distribute individual tracks without express permission from Clarion Magazine.

Production notes and Skype suggestions

A number of listeners have asked how we record Planet Clarion. While all anyone really needs to record a podcast is a microphone, a PC, and some recording software, our setup is a little bit more complicated, mainly because we (Andrew and Dave) live 1700 miles apart. It's all made possible by Skype, a free Internet phone service, which we highly recommend. The human ear can detect sounds in the frequency range of about 50 Hz (cycles per second) to 20,000 Hz; Skype transmits frequencies from 50 to 8000 Hz, which is pretty much the range of human speech, as compared to the plain old telephone service (POTS) which has a relatively narrow range of 300 Hz to 3300 Hz.

Dave records the Skype conversation on his PC, in two tracks, using Sony Vegas; one track is Dave's microphone, and the other is Andrew via Skype. We originally intended to record Andrew's microphone on his PC as well, but the Skype recording was good enough that we just ran with that. The two-track approach makes it possible to compensate for differences in microphones, sound quality, and sound levels. Among other tools, Dave uses the Endorphin plugin (via the VST-DX Wrapper, since Vegas does not support VST plugins natively).

We both use headsets rather than standalone microphones. Dave is geeked up with an AKG HS200, and Andrew uses an Altec Lansing AHS 502, which is a reasonably-priced, good quality headset. We recommend the Altec Lansing products for anyone wanting to get started with Skype. If you're not sure which is the best Altec Lansing (or other manufacturer's) model in your area, find out what the local gamers like to use, and you'll probably be in good shape. Make sure whatever you buy will work with your sound card - if you want to go high end, it's more likely that your headset will need a pre-amp between you and the sound card. Also while you can use Skype with a microphone and speakers, we don't recommend that setup. It's too easy for the microphone to pick up sound from the speakers. You may think everything's fine, but the person at the other end will hear an echo. A headset removes this problem, and it also keeps the microphone a constant distance from your mouth, giving you more freedom of movement (especially if it's a wireless headset).

If you are considering a USB headset, keep in mind that these sometimes have greater latency than a regular headset/sound card combination. There's always some delay between the time a signal is generated by the microphone and the time the hardware finishes processing, and of course it also takes time to transmit the signal to the person at the other end of your Skype conversation. You want to keep latency to a minimum.

Quick Links

HOME

Subscribe/Renew

Clarion Books

FAQ (Frequently Asked Questions)

Log In

Download PDFs

Free Membership

Free Articles

Advertise

Reader Comments

Write For ClarionMag

Privacy

Contact

Clarion Magazine

Calling the Skype API

by David Harms

Published 2004-12-22

If you listened to the December 15 Planet Clarion podcast, you heard the conversation Andrew Guidroz II and I had with Colin Wynn about integrating Skype with a Clarion application. Skype, if you haven't heard, is high quality, peer to peer Internet telephony software. You can talk to anyone anywhere in the world for free – all you both need is the Skype software (also free), a headset (or microphone and speakers) and a decent (preferably high speed) Internet connection. If you want to talk to someone who doesn't have Skype, you can use the SkypeOut service, at very reasonable rates, although the sound quality drops sharply to that offered by POTS (Plain Old Telephone Service).

When Colin first told me he was embedding a Skype link in his Clarion apps, I assumed he was using the Skype API. This is an interface that lets other applications talk to, and control, the Skype client. Instead, Colin was doing something quite simple and elegant – he was using a callto: HTTP link just the way you'd use a mailto: link.

Still, I was curious about the Skype API. It would be very cool to have a Skype instant message autoresponder, or an answering machine, or any of a dozen other Skype-related utilities. After a little research, including a helpful newsgroup message by Larry Sand on the subject of windows messages, I was off to the races.

The Skype API

At present, there are three main sources of information about the Skype API. One is the API documentation itself, which although concise, reminds me a lot of the Clarion 2.0 documentation. The information is there, but it's not all that reader-friendly. The second source is a C++ example application, with source code, and the third is the Skype API forum.

My first step was to convert the C++ example application to a Clarion equivalent, but before I show how that's done let me give you a brief overview of how the API works.

Interprocess communication

You don't need a LIB to use the Skype API, because there is no DLL that contains the API functions. Instead, your application communicates with the Skype client by posting and receiving text messages, encoded as UTF-8. You send messages using the `SendMessage` Windows API call, and you receive messages by subclassing one of your application's windows and listening for the replies.

The API documentation is simply a description of what all of these messages are, and mean. With the API you can:

- Establish and monitor a connection with the Skype client
- Search the contacts database
- Monitor changes to the contacts database
- Search the instant message database
- Search the calls database
- Determine Skype's audio device configuration
- Initiate and answer phone calls
- Monitor the status of phone calls
- Send and receive instant messages
- Open dialogs
- and much more

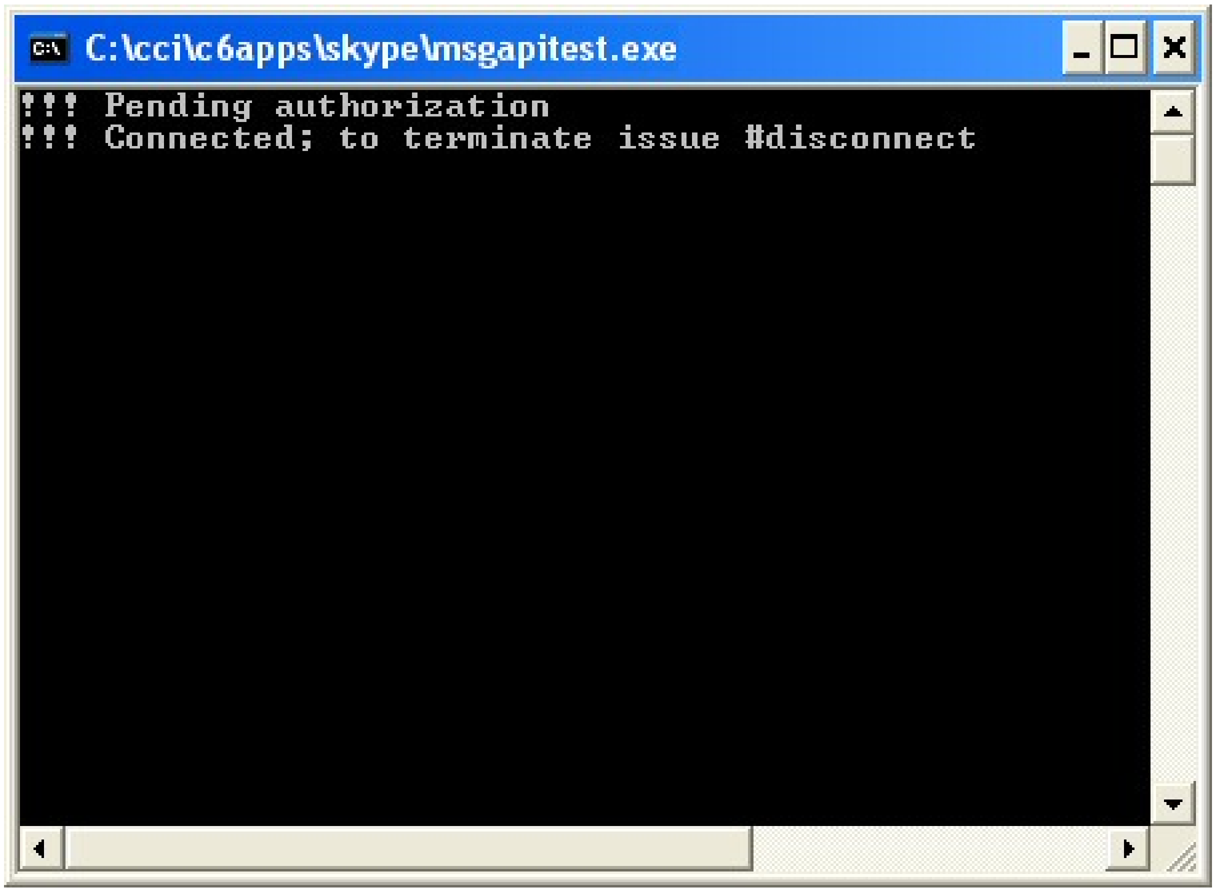
But what about this requirement that all calls to the API be UTF-8 encoded? For just sending commands to the API, you really don't need to worry about this, since the first 128 ASCII/ANSI characters are the same in UTF-8. From MSDN:

> UTF8 is a code page that uses a string of bytes to represent a 16-bit Unicode string where ASCII text (<=U+007F) remains unchanged as a single byte, U+0080-07FF (including Latin, Greek, Cyrillic, Hebrew, and Arabic) is converted to a 2-byte sequence, and U+0800-FFFF (Chinese, Japanese, Korean, and others) becomes a 3-byte sequence.

What this means is that as long as you're just talking to the API, or sending normal English text, you won't have a problem. Otherwise you'll probably want to use the API call `MultiByteToWideChar` to convert the text to Unicode, then `WideCharToMultiByte` to convert the Unicode to UTF-8. I haven't tried this yet, however.

The example program

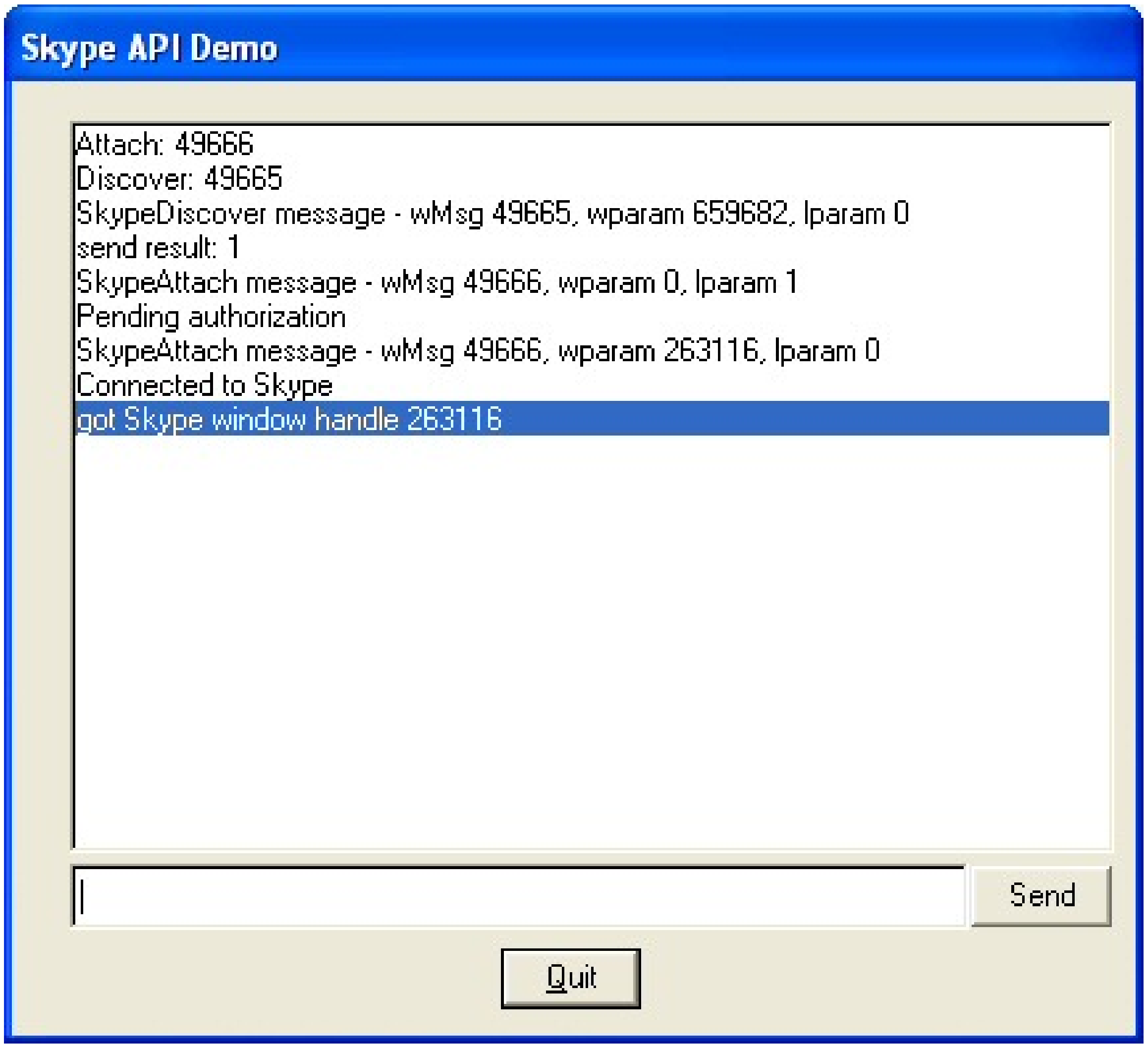
For the remainder of this article I'll explain the Clarion version of the example app that comes with Skype. Figure 1 shows the Skype C++ app in action. As you can see, this is a console application.



```
!!! Pending authorization
!!! Connected; to terminate issue #disconnect
```

Figure 1. The Skype API example application

Figure 2 shows the Clarion equivalent of the C++ application.



```
Attach: 49666
Discover: 49665
SkypeDiscover message - wMsg 49665, wparam 659682, lparam 0
send result: 1
SkypeAttach message - wMsg 49666, wparam 0, lparam 1
Pending authorization
SkypeAttach message - wMsg 49666, wparam 263116, lparam 0
Connected to Skype
got Skype window handle 263116
```

Figure 2. The Clarionized Skype API example application

There are a few important points to note about this app. First, this is not intended as an example of Skype programming style, not by me, and not, most likely, by the authors of Skype. All it is intended to do is demonstrate how to exchange messages with the Skype client. You'd probably want something much more extensible, say a nice class to wrap up all the important functionality. (And in fact there is at least one Skype COM control already available.)

As you can see from Figure 2, the Clarion version of the example app is a simple window, with a list box to display the messages sent and received, an entry field where you can enter commands, and a Send button. and a Quit button.

The MAP

This application uses a handful of API calls. The MAP is as follows:

```
map
   Module('WINAPI')
      RegisterWindowMessage(*CSTRING MsgStr),ULONG,|
         PASCAL,RAW,NAME('RegisterWindowMessageA')
      SendMessage(ULONG hWnd,ULONG uMsg, ULONG wParam, |
         LONG lParam),LONG,PASCAL,NAME('SendMessageA')
      CallWindowProc(LONG,UNSIGNED,SIGNED,UNSIGNED,LONG),|
         LONG,PASCAL,NAME('CallWindowProcA')
   END
   MODULE('Clarion')
     MemCpy(LONG lpDest, LONG lpSource, LONG nBytes), |
        LONG, PROC, RAW, NAME('_memcpy'),proc
   END
   SkypeMsgHandler(ulong hwnd, ulong wMsg, ulong wparam, |
      LONG lparam),LONG,PASCAL
   Show(String Msg)
END
```

The `RegisterWindowMessage` converts any string into a message identifier. When the app starts up, one of the first things it does is register two message names, `'SkypeControlAPIAttach'` and `'SkypeControlAPIDiscover'`. The first time `RegisterWindowMessage` receives a given string, it assigns a message ID that is guaranteed to be unique system-wide. Subsequent calls for that string return the assigned message ID. This way both Skype and this application can communicate by broadcasting messages using that ID.

The `CallWindowProc` message is needed to install the message handling procedure (`SkypeMsgHandler`) and `MemCpy` is a memory copy function that helps with receiving a copy of the original message, as I'll explain later. The only other function in the map is `Show`, which displays any received string in the list box.

The data

There are a couple of data declarations you'll need. The first is `CopyDataStruct`, which declares the format of messages passed by SendMessage.

```
COPYDATASTRUCT          GROUP,TYPE
dwData                     LONG
cbData                     LONG
lpData                     LONG
                        END
```

`dwData` is any 32 bit value, `lpData` is a pointer to a block of data that is being passed, and `cbData` is the length of the data `lpData` points to.

There are a couple of Windows equates you'll need to know:

```
WM_COPYDATA             EQUATE(04Ah)
HWND_BROADCAST          EQUATE(0FFFFH)
```

And there are some Skype equates:

```
SKYPECONTROLAPI_ATTACH_SUCCESS               EQUATE(0)
SKYPECONTROLAPI_ATTACH_PENDING_AUTHORIZATION EQUATE(1)
SKYPECONTROLAPI_ATTACH_REFUSED               EQUATE(2)
SKYPECONTROLAPI_ATTACH_NOT_AVAILABLE         EQUATE(3)
SKYPECONTROLAPI_ATTACH_API_AVAILABLE         EQUATE(8001H)
```

There are a few other variable declarations, but nothing that needs any explanation.

The code

The first step, after opening the window, is to install the message handler:

```
origMsgHandler = window{prop:WndProc}
window{prop:WndProc} = address(SkypeMsgHandler)
```

This is what is generally called subclassing, which is a confusing term because it's not subclassing in an OOP sense. Normally, Clarion handles most

Windows messages automatically, and only passes along, via the `ACCEPT` statement, those messages likely to be useful to the Clarion developer. If you want to get access to other messages, such as those broadcast by Skype, you have to install your own message handler, which will process those messages before passing control back to the original message handler (which is why you save the address of that internal function using `prop:WndProc`).

Now it’s time to get the system IDs of the standard Skype messages:

```
Msg = 'SkypeControlAPIAttach'
MsgSkypeAttach = RegisterWindowMessage(Msg)
Msg = 'SkypeControlAPIDiscover'
MsgSkypeDiscover = RegisterWindowMessage(Msg)
```

RegisterWindowMessage takes a pointer to a `CSTRING`, which is why I have to load a variable with the value and then pass the variable.

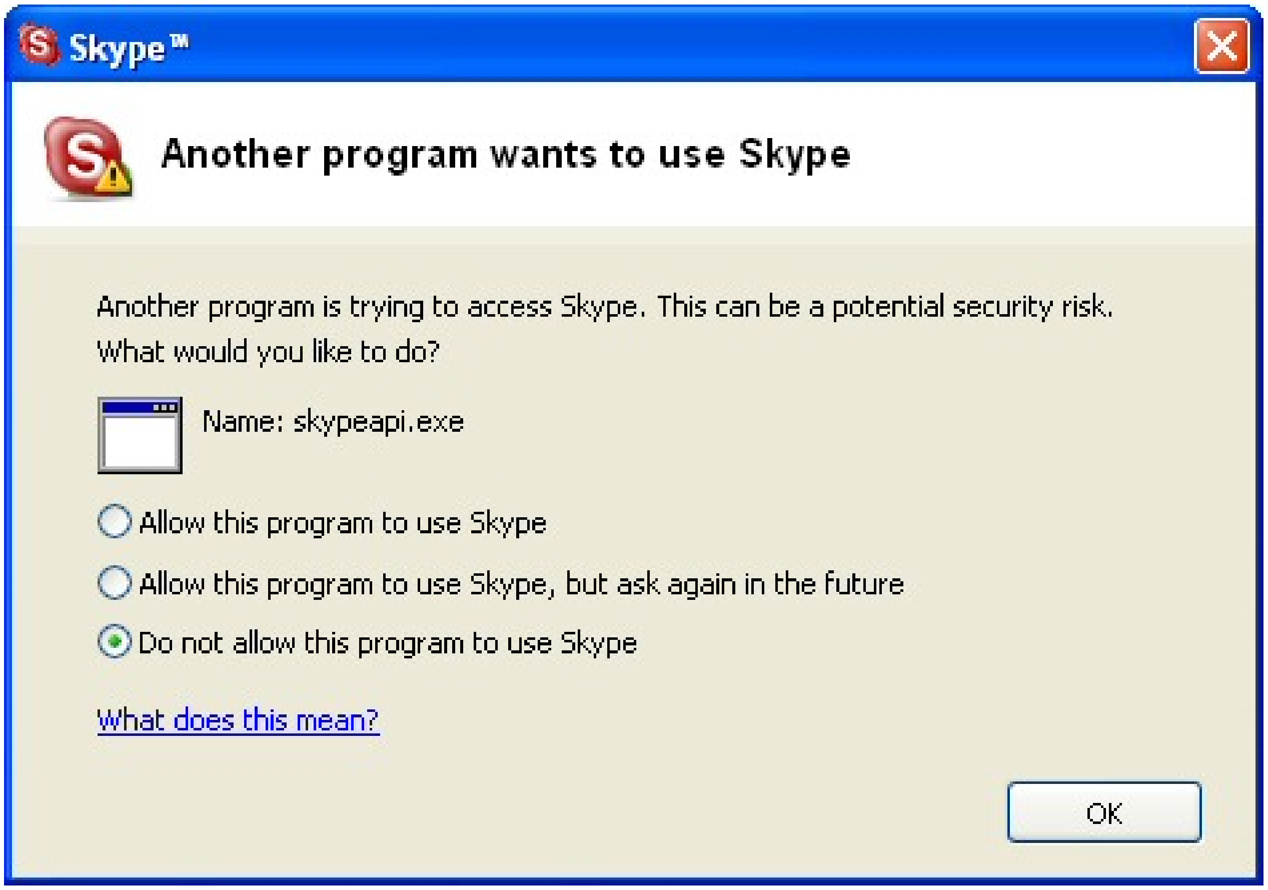
Next, broadcast `MsgSkypeDiscover`, which tells Skype, if present, to respond:

```
result = SendMessage( HWND_BROADCAST, MsgSkypeDiscover, |
   window{prop:handle}, 0)
```

If Skype is running, and sees this message, it will respond back with a `SkypeControlAPIAttach` message. The possible values are, as listed in the equates above:

```
SKYPECONTROLAPI_ATTACH_SUCCESS
SKYPECONTROLAPI_ATTACH_PENDING_AUTHORIZATION
SKYPECONTROLAPI_ATTACH_REFUSED
SKYPECONTROLAPI_ATTACH_NOT_AVAILABLE
SKYPECONTROLAPI_ATTACH_API_AVAILABLE
```

The first step is authorization. When you attempt to connect, Skype pops up a message like the one in Figure 3.


Skype™

Another program wants to use Skype

Another program is trying to access Skype. This can be a potential security risk.
What would you like to do?

Name: skypeapi.exe

Allow this program to use Skype [ ]

Allow this program to use Skype, but ask again in the future [ ]

[✓ Do not allow this program to use Skype]

What does this mean?

Figure 3. Skype authorization request

If you allow the program to access Skype, you’ll get an Attach Success message, otherwise it will be an Attach Refused message. I’ll show how the example application receives these messages shortly. If you block the application it will stay blocked, until you go to File|Options in Skype, and under the Privacy tab click on Manage other programs’ access to Skype and remove the corresponding block entry.

Assuming you’ve allowed the example app to connect, you’re now ready to send and receive commands. When you enter some text in the command field and press Send, the following code executes:

```
SendData.dwData = 0
SendData.cbData = Len(cmd)+1
SendData.lpData = Address(cmd)
IF (SendData.cbData > 1)
   result = SendMessage(SkypeWinHandle, WM_COPYDATA, |
      window{Prop:Handle}, Address(SendData))
   show('sending to ' & SkypeWinHandle & ', result: ' & result)
```

```
END
```

Sending data is simply a matter of setting `dwData` to 0 (an arbitrary value specified by the Skype API, as far as I can tell), `cbData` to the length of the command `CSTRING`, and lpData to the address of the command `CSTRING`, and then calling `SendMessage`. `Show` displays the sent message in the list box.

Receiving messages

Receiving messages from Skype is the trickiest part, and it's not all that difficult. As I said, you need to install a message handler procedure, which I've called `SkypeMsgHandler`.

`SkypeMsgHandler` first examines the second parameter it receives to see if it corresponds to one of three values: `WM_COPYDATA`, `MsgSkypeAttach`, or `MsgSkypeDiscover`.

Only the first two really matter – I put in some code to display the discovery message, but it isn't necessary.

As you would guess, `MsgSkypeAttach` only comes into play when you're attempting to connect to Skype. And other than reporting errors back to the user, there's only one crucial bit of code here:

```
case lParam
of SKYPECONTROLAPI_ATTACH_SUCCESS
   SkypeWinHandle=wparam;
```

You must store the `wParam` value passed with this message, as it contains the handle of the Skype window where you will send all subsequent messages.

After processing a Skype message the subclassing procedure must return a non-zero value or Skype will consider the connection broken. I haven't had this happen with the API connection messages, but it will definitely happen with `WM_COPYDATA`

And speaking of WM_COPYDATA, here's the code for receiving all other Skype messages:

```
of WM_COPYDATA
   if SkypeWinHandle=wParam
     MemCpy(Address(ReceiveData),lParam,Size(ReceiveData))
      TempData.CDSType = ReceiveData.dwData
      If ~(TempData.Message &= Null)
          Dispose(TempData.Message)
      END
      TempData.Message &= New(CString(ReceiveData.cbData + 1))
      MemCpy(Address(TempData.Message),|
         ReceiveData.lpData,ReceiveData.cbData)
      show(TempData.Message)
      return 1 ! any non-zero value
   END
```

You must make a copy of the received message, since its memory was allocated by another program. Do this with the memcpy function. All this app does is display the message – you would probably want to actually do some processing in response to the message. Again, you must return a non-zero value after processing the WM_COPYDATA message.

Points of interest

There are a few points to note about the Skype API. One is that your application must not take more than one second to respond to any message or Skype will consider the connection broken. So if you have anything lengthy to do in response, don't do it inside the message handler.

Another is that the connection between your application and Skype is based on Windows user information, at least on W2K and Windows XP. For instance, let's say you run Skype and the example application, and they establish communication. Next, right-click on the Skype short cut, select Run As, and run a second instance of Skype as another user. Do the same with a second instance of the example application, also running as that user. You will now have two copies of the sample app, and two copies of Skype, and only the like-user copies will communicate with each other.

Running the example app

This application is not an APP, but source code with a PRJ. To run it, copy the zip contents to a directory, and in Clarion use Project|Set to specify the project file. Then do a make and run as usual.

Here are a few commands you can try:

Command	**Response**	**Notes**
PING	PONG	Verifies the connection is open
MESSAGE *user text*	MESSAGE *id* STATUS SENDING MESSAGE *id* STATUS SENT *or error messages*	Send an instant message – *id* is the internal message ID
CALL *userid*	CALL *id* STATUS ROUTING CALL *id* STATUS RINGING CALL *id* STATUS INPROGRESS CALL *id* STATUS	Call a Skype user – *id* is the internal call ID.

	DURATION *time* CALL *id* STATUS FINISHED *or error messages*	

Conclusion

The more I use Skype, the more intrigued I am by the opportunities opened up by high-quality, free internet telephony. The Skype site lists a few possibilities, including personal information managers, enhanced gaming audio, and on-demand content and interactive voice response (IVR) applications. I hope this translation of the Skype API example will open up new possibilities for your Clarion development.

Download the source

David Harms is an independent software developer and the editor and publisher of Clarion Magazine. He is also co-author with Ross Santos of Developing Clarion for Windows Applications, published by SAMS (1995), and has written or co-written several Java books. David produces the Planet Clarion podcast, which he co-hosts with Andrew Guidroz II.

Reader Comments

Add a comment

Very cool dave thanks

Quick Links

HOME

Subscribe/Renew

Clarion Books

FAQ (Frequently Asked Questions)

Log In

Download PDFs

Free Membership

Free Articles

Advertise

Reader Comments

Write For ClarionMag

Privacy

Contact

Clarion Magazine

Using Arrays in Databases

by Dermot Herron

Published 2004-12-23

Arrays in databases? You *can't* in SQL and you *shouldn't* in TPS!

So why bother with this article? These are some techniques that make a database *look* as though it has arrays and, in hand-coding, these arrays can be used very efficiently. (The approach I will describe does not work with the templates!)

Imagine you actually had arrays in SQL and *easy* arrays in a browse box. Then you might declare the following fields in the database (amongst many others):

```
Date       LONG                ! Date of appointment
Time       LONG                ! time of appointment
BookedName STRING(30),DIM(4) ! 4 columns in which people may book
BookDate   LONG, DIM(4)      ! date when the booking was made
BookTime   LONG, DIM(4)      ! time when the booking was made
```

and have a browse box with the columns:

```
Date Time BookedName[1] BookedName[2] BookedName[3] BookedName[4]
```

You could record the date and time the booking was made in `BookDate` and `BookTime`. You could then process the columns in a `LOOP` because they are indexed (`BookedName[Cntr]`) instead of using many `CASE` statements to handle `BookedName1`, `BookedName2` etc.

It is actually necessary in a Browse box (and a database) to put real fields in the columns, but this makes it difficult to do any calculations on these fields because they have to be individually named. (You *can* put arrays into browses etc. but you cannot populate them in the AppGen from the OVER except purely by hand, which I found to be very clumsy – see *Array and Complex Structure Field Equates* in the Help)

Clarion has a brilliant data technique, derived from C, which allows you to make one piece of data *look like another* called OVER. Using an OVER is like using an overlay to mark multiple-choice questions easily – it shows the data "through holes" to speed up the operation. Note that the *same* data is visible – you just *see* it differently because of the OVERlay.

There are two techniques that I use that apply to databases. One is used to index browser-displayable fields for quick accurate access, and the other allows storage of arrays that are not expected to be "browsed", in a non-array database. The sample app I use to demonstrate these techniques is thanks to Barry Pratt - he has written a golfing rounds analysis program using my array approach.

Indexing a set of browse-displayed fields:

Imagine you need to display golf-scores and other details for 18 holes. (I *know* you should put 18 records into a separate related database and fill a queue etc, but this makes a good example, and does have uses in a real program) Each record in the database is *one round* of golf. Then you might have a database looking like this, with 18 values of each in a database record:

```
Rounds       FILE,DRIVER('Topspeed'),|
               NAME('Rounds.tps'),PRE(Rnd), |
               CREATE,BINDABLE,THREAD
ByDate           KEY(Rnd:Date)
Record           RECORD,PRE()
Date               LONG
Description1       STRING(20)
```

```
Description2        STRING(20)
Description3        STRING(20)
…
Description18       STRING(20)
Par1                SHORT
Par2                SHORT
Par3                SHORT
…
Par18               SHORT
ShotsToGreen1       SHORT
ShotsToGreen2       SHORT
ShotsToGreen3       SHORT
…
ShotsToGreen18      SHORT
Putts1              SHORT
Putts2              SHORT
Putts3              SHORT
…
Putts18             SHORT
TotalShots1         SHORT
TotalShots2         SHORT
TotalShots3         SHORT
…
TotalShots18        SHORT
            END
        END
```

Because the fields are separate, they can be displayed in a browse without difficulty (tedious!).

You *have* to have separate fields *in the database* if it is SQL, and anyway it is short-sighted to use arrays even in a Topspeed database. (You shouldn't use arrays in Topspeed in case you later change to SQL, and also because there have been lots of bugs in the TPS array handling in earlier Clarion versions.)

If you now code this in the conventional manner, you have to do something like this:

```
LOOP Cntr = 1 TO 18
  CASE Cntr
  OF 1
    TotalShots1  =  ShotsToGreen1  + Putts1
  OF 2
```

```
    TotalShots2  =  ShotsToGreen2  + Putts2
  OF 3
    TotalShots3  =  ShotsToGreen3  + Putts3
  …
  OF 18
    TotalShots18 =  ShotsToGreen18 + Putts18
  END
END
```

And how easy it is to forget to update one entry of a copy-paste, like this:

```
TotalShots3 = ShotsToGreen2 + Putts3
```

It's difficult to notice a bug like this. Wouldn't it be much less error-prone and *much* easier if you could write:

```
LOOP Cntr = 1 TO 18
  TotalShots[Cntr] =  ShotsToGreen[Cntr] + Putts[Cntr]
END
```

You *can*, if you use an OVER.

Note that the entire record is OVERed because you have to have "spacers" (the oDate) to line up the arrays. To avoid errors *always* put in the complete record. The size of the OVER must be identical to the size of the RECORD (see the end for a precautionary test for this). I like to start my OVER variables with "o" to keep very clear exactly what I am using.

In the global, AfterFileDeclarations (so there is something for this group to *be* OVER) you would put this code:

```
oRnd           GROUP, OVER(Rnd:Record)
oDate            LONG
oDescription     STRING(20),DIM(18)
oPar             SHORT,DIM(18)
oShotsToGreen    SHORT,DIM(18)
oPutts           SHORT,DIM(18)
oTotalShots      SHORT,DIM(18)
               END
```

(I try not to use a PRE because it is going out of fashion. The "dot" syntax is preferred.)

The code to process this now looks like:

```
! code for fetching the record from the database goes here
!
LOOP Cntr = 1 TO 18
  oRnd.oTotalShots[Cntr] =  oRnd.oShotsToGreen[Cntr] |
    + oRnd.oPutts[Cntr]
END
!
! code for UPDATEing the record goes here
```

and this actually *works!*

Storing arrays that will *not* be used in a browse in a database

It can be very convenient to store array-type data in a database, because it saves much typing and error-prone repetition when setting up the database. It also speeds up generation (surprisingly) because the generator doesn't like tables with many fields as much as simple tables.

This can be done by using a STRING or CSTRING the correct size to match the array. Remember that the size limits of a modern database are huge (String *Field* Size : 15,000 bytes in TopSpeed, and maybe one could use a memo or blob but I haven't tried this).

So a database might be defined as follows:

```
DemoData       FILE,DRIVER('Topspeed'),NAME('DemoDat.tps'),|
                        PRE(Dmo), CREATE,BINDABLE,THREAD
ByDate           KEY(Dmo:Date)
Record           RECORD,PRE()
Date               LONG
LongString         CSTRING(40)   ! 4  bytes per 10 LONGs
ShortString        CSTRING(20)   ! 2  bytes per 10 SHORTs
StringString       STRING(150)   ! 15 bytes per 10 STRINGs
```

```
RealString         CSTRING(40)   ! 4  bytes per 10 REALs
                 END
               END
```

And this would have an OVER that looks like

```
oDmo        GROUP, OVER(Dmo:RECORD)
oDate         LONG                ! the usual spacer
oLong         LONG,DIM(10)        ! OVER LongString
oShort        SHORT,DIM(10)       ! OVER ShortString
oString       STRING(15),DIM(10) ! OVER StringString
oReal         REAL,DIM(10)        ! OVER RealString
            END
```

Now, of course, you can refer to the record by an index:

```
LOOP Cntr = 1 TO 10
    oDmo.oReal[Cntr]   = oDmo.oLong[Cntr] / oDmo.oShort[Cntr]
    oDmo.oString[Cntr] = FORMAT(oDmo.oDate, @D5) & ' ' &
                             FORMAT(oDmo.oReal[Cntr], @N8,2)
END
```

Note the use of `oDmo.oDate`. You could use `Dmo:Date` which is exactly the same thing but it would become *very* difficult to see what the code is doing.

The downside of storing arrays in a database

This technique isn't without problems – it totally confuses the TopScan utility. You can look at an arrayed table *once* and thereafter TopScan cannot cope. You have to go and delete the `<filename>.scn` that TopScan generates to look at the database again. I think the funny characters in string data confuse the program.

Even if you do view the database with TopScan, the data in the DIMmed fields is visual junk. You are looking at a REAL through the eyes of a STRING. So if looking at the data is important, you have to write a program to make the data visible.

Protecting yourself from errors

It is very easy to add a field to the database and forget to add it to your OVER, so now the two no longer match up. This is a very difficult bug to find. So *always* automatically protect yourself *whenever* you put in an OVER by putting this code in the EnterProcedureScope embed of the frame or first window:

This won't catch every possible error but it will catch the common mistakes.

```
IF SIZE(Dmo:RECORD)  ~=  SIZE(oDmo)
    MESSAGE('Programmer - SIZE(Dmo:RECOR)  ~=  SIZE(oDmo)')
    HALT()
END
```

Initializing the database

If you use a STRING in the TopSpeed driver, it is cleared to spaces on inserting a new record. This leaves *every* BYTE of the STRING filled with 32 (VAL('<space>')), which gives peculiar numbers if OVERlaid with a REAL or other number. The best way to fix this is to use a CSTRING where it is overlaid with numeric variables. Topspeed clears a CSTRING to all \0 or ASCII[0] on inserting. Note that you do not have to allow the extra character for the usual \0 at the end because you are never using the data as a string.

Otherwise use CLEAR(oDmo) if an INSERT is happening. This will genuinely *zero* the REAL, LONG etc and not just zero the string to spaces. This must be done in PrimeRecord. (Be careful if you are using an auto-incrementing key because you can clear that by mistake if you do this in the wrong place!)

If you are using SQL, you *have* to use the CLEAR(oDmo) method, because a CSTRING (in MSSQL at least) has a \0 in the first location and garbage in the rest of the string.

More complex OVERs (not for the faint-hearted)

Clarion allows extremely complex group and arrays of groups which can be used as OVER. One could alternatively lay out the Rounds database thus:

```
Rounds          FILE,DRIVER('Topspeed'),NAME('Rounds.tps'),|
                   PRE(Rnd), CREATE,BINDABLE,THREAD
ByDate            KEY(Rnd:Date)
Record            RECORD,PRE()
Date                LONG
Description1        STRING(20)
Par1                SHORT
ShotsToGreen1       SHORT
Putts1              SHORT
TotalShots1         SHORT
Description2        STRING(20)
Par2                SHORT
ShotsToGreen2       SHORT
Putts2              SHORT
TotalShots2         SHORT
Description3        STRING(20)
Par3                SHORT
ShotsToGreen3       SHORT
Putts3              SHORT
TotalShots3         SHORT
…
Description18       STRING(20)
Par18               SHORT
ShotsToGreen18      SHORT
Putts18             SHORT
TotalShots18        SHORT
                  END
                END
```

And here the OVER to use is

```
oRnd              GROUP, OVER(Rnd:Record)
oDate               LONG
oHol                GROUP,DIM(18) ! it is the GROUP that is DIMmed
oDescription          STRING(20)
oPar                  SHORT
oShotsToGreen         SHORT
oPutts                SHORT
oTotalShots           SHORT
```

```
                    END
                  END
```

Note that the "shape" of this mirrors the layout of the database with the repeated GROUP of fields for each hole DIMmed.

And the code that uses this is:

```
! code fetching the record from the database goes here
!
LOOP Cntr = 1 TO 18
  oRnd.oHol[Cntr].oTotalShots =  |
    oRnd.oHol[Cntr].oShotsToGreen + oRnd.oHol[Cntr].oPutts[Cntr]
END
!
! code UPDATEing the record goes here
```

The syntax of the variables is complex, but if you read it carefully, it has a very clear logic.

The GROUP (oHol), inside the main OVER GROUP (oRnd) is repeated, so it gets the index, not the variable itself:

```
 oRnd.oHol[Cntr].oTotalShots
```

I have even used OVER over the browse queue in order to process the columns as an array to set colors. Once you are comfortable with OVER on a database, you will find there are many other areas that become simpler and easier using an OVER.

Conclusion

It is possible to use apparent arrays in databases including SQL, even if SQL doesn’t let you use real arrays! It is more complex to code and has to be handled by hand. Of course, the circumstances dictate if it is worth making the data into an array. If you have only three things you want arrayed, a CASE statement looks easier. I developed my OVER techniques because I had

20 columns each with 10 additional stored pieces of data, and it would have become unwieldy to use anything else.

Download the source

Dermot Herron grew up and went to school in Rhodesia, and earned an electrical-engineering degree at Capetown University. In the 1970s he worked for the Rhodesian Post Office, which was then also the telephone company. He was given the use of an HP9100, which was the very first programmable desktop computer, to help with the design of party-line telephones. He totally fell in love with computers then and there. Leaving the Post Office, he traveled for four years, worked in Canada (as a cabinet maker) and in England (as a microprocessor engineer) and then immigrated to New Zealand. But Dermot learned that once Africa gets into your blood you are doomed! Now he runs his own company in Johannesburg, writing software to send messages to mobile phones. He lives on a 19-acre plot with a river, and spends his off-time fixing things.

Reader Comments

Add a comment

Good Article I use this technique quite a bit. If you...

Quick Links

HOME

Subscribe/Renew

Clarion Books

FAQ (Frequently Asked Questions)

Log In

Download PDFs

Free Membership

Free Articles

Advertise

Reader Comments

Write For ClarionMag

Privacy

Contact

Clarion Magazine

Clarion News

Search the news archive

Boxsoft Development Named Third-party Vendor of the Week

Mke Hanson and Boxsoft Development have been named by the Clarion Developers Challenge Football Contest as the Third-party Vendor of the Week. Mike and Boxsoft Development are awarding this week's winner of the Clarion Developers Challenge Football Contest a copy of their Super Templates (winner's choice), the easy way to add Super functionality to your Clarion applications.

Posted Wednesday, December 22, 2004

Product Scope 32 PRO Special

Product Scope 32 PRO, Version 5.0 Single User Spreadsheet license is on special. For a limited time, the price is USD $14.95 each for 1 to 5 copies. Site licenses can be bought at this special price as well. This program is used to develop, maintain, and display the Clarion Third Party Profile Exchange. Version 5.0 introduced a number of new tools for Profile Exchanges (Product and Vendor Information) and general utility work for image files, PDF files, and file utility work such as copy, rename, move, launch, etc.

Posted Wednesday, December 22, 2004

vuFileTools 3.0

vuFileTools version 3.0 has been released. Order vuFileTools by 12-31-2004 and receive a copy of vuAgent (interface to MS Agents Merlin, Peedy, etc) free (being released this week as well).

Posted Wednesday, December 22, 2004

Encode Your Email Links

This email link encoder makes it more difficult for spammers to harvest your email address.

Posted Wednesday, December 22, 2004

New Softvelocity Drivers Available

You can now purchase the SoftVelocity IP Driver ($300), In-Memory Driver ($225), and Dynamic File Driver ($225) from CapeSoft. For a limited time only you can get all three drivers together for $675. Other bundles are also available.

Posted Wednesday, December 22, 2004

CapeSoft Price Increase Looming

As of January 1st, there will general price increase across the CapeSoft Product range. If you are thinking of adding some CapeSoft products to your Clarion toolbox, now is the time to get them.

Posted Wednesday, December 22, 2004

File Manager 3 Sybase Support

FM3 now supports a total of 7 Database Drivers, and 5 popular SQL Backends. These drivers include TopSpeed, Clarion, Btrieve, MsSql, Oracle, ODBC and Sybase, and includes MsSql, Oracle, MySQL, FireBird and Sybase Backends. With the growing number of database support comes a brand new ConnectToSQLBackend Control Template which is far more fool proof, generic and customizable than before. Be sure to check out a complete set of new SQL Examples which now show far more real to life SQL scenarios in terms of Database Design. Conversion from TPS to SQL, RelationShips in the Dictionary, and examples of each supported Datatype. FM3 now also supports the brand new IP Driver released recently by SoftVelocity (and available for purchase online at ClarionShop). FM3 costs $249.

Posted Wednesday, December 22, 2004

GPF Reporter

CapeSoft's GPF Reporter is tool that not only tells you that a GPF happened, but also tells you the code sequence executed prior to the GPF, and reports the line number the GPF occurred on. Compatible with Clarion 5 and up, ABC and Legacy, Multi-DLL, and Standalone and Local Compile modes. GPF Reporter costs $99.

Posted Wednesday, December 22, 2004

Secwin Online Server!

Secwin Online Server provides immediate access to temporary or permanent product activation codes via the internet, for any product which has the Secwin Online Client enabled. Activation codes can be blocked for specific clients, products or datasets, allowing the supplier full control over product sales and distribution, but giving prospective or new clients immediate access to products. Activation codes can be issued as temporary or permanent product codes. Secwin Online Server features: Can be installed as a service; Creates Secwin Activation codes for any product; Authorize temporary or permanent activation codes; Features a web user interface; Datasets, clients or products can be blacklisted, preventing any further issues of activation codes; Stores product and client activation code history; Requires Windows 98se or higher operating systems. Secwin Online Server requires your Client Applications to use Capesoft SecWin and NetTalk. Secwin Online Server is being released at the special price of $199 during it's Beta phase. The expected Gold price will be $299.

Posted Wednesday, December 22, 2004

CPCS v6.11 For C6 9031

CPCS v6.11 and all related add-ons have been rebuilt with hotfix #9031 and are now available for download.

Posted Wednesday, December 22, 2004

cpTracker Sale & News

Through the end of 2004, cpTracker Pro is now on sale for $79.00 per user (reg $99). cpTracker Lite is on sale for $19.00 per user (reg $27). You can get the Clarion DCT for an extra $15 (reg $25). cpTracker Pro is a CRM/Contact, Project, Task, Sales and HelpDesk Management.

Posted Wednesday, December 22, 2004

RPM & AFE for C6 9031

Updated installs for RPM and AFE are available for C6.1 9031.

Posted Wednesday, December 22, 2004

dpQuery 2.02 Full Version

The full version of dpQuery 2.02 Is now available. New features include: Binary data import; Import form XML data source; Possibility to edit/delete records from Data queue before final importing; PreProcessField virtual method to get data queue filed number and data buffer size; Possibility to assign Description with the text from "Field description" or "Column heading" automatically. Changes include: SetDescription virtual method; Templates in order to work with the Binary data importing; Some variables and methods name to avoid the names conflict.

Posted Wednesday, December 22, 2004

Clarion Developers Challenge Week 14 Winner

First place for Week 14 of the Clarion Developers Challenge goes to Bob Foreman. Bob finished in a five-way tie with Alejandro Contreras, Kelly Major, Ramon Reed and Dean Burgess in this week's contest, each having a total of 10 correct picks. Bob then pulled out the win in the Monday night tiebreaker by picking a tiebreaker score of 55 (actual score was 87). For finishing first in this week's Clarion Developers Challenge, Bob will be receiving a copy of the PD Browse Button Lookup, donated by Phil Will and ProDomus Software.

Posted Wednesday, December 22, 2004

IceTips MySetups

Icetips Software has released its first non-Clarion specific software: MySetups. MySetups is designed to keep track of information about installation or setup files. The program keeps track of the actual files as well as information about where the install was downloaded from, login information to websites, keycodes, serial numbers and any number of additional information you want to add. MySetups keeps track of who is the registered owner, who is the producer of the software as well as support emails, websites etc.

Posted Wednesday, December 22, 2004

dpQuery 2.02 Demo

A new demo of dpQuery 2.02 is now available, and the full release is coming soon. New features include: Binary data import; Import form XML data source; Possibility to edit/delete records from Data queue before final importing; PreProcessField virtual method to get data queue filed number and data buffer size; Possibility to assign Description with the text from "Field description" or "Column heading" automatically.

Posted Wednesday, December 15, 2004

vuFileTools Version 3 Quick Reference

The vuFileTools version 3 quick reference guide is available online. All functions that are new to version 3 are highlighted in red.

Posted Wednesday, December 15, 2004

BST 3.702

BST 3.702 has been released. This version includes a bugfix to Prop:SQL in the Wall Calendar when limited to one resource selected. A cumulative patch for Ver 3.70 or 3.701 is available.

Posted Wednesday, December 15, 2004

SELECT Class

Carlos Gutierrez has written a small class to make it easier to create ad-hoc SQL queries. This class is released as copyrighted freeware source. Features include: Easy to convert QA queries to Clarion; Converts SQL dates to Clarion Dates; Columns available by column name, e.g. D.COL('ColumnName'); Query result can be assigned to a group or queue, e.g. D.DEEP(GroupName).

Posted Wednesday, December 15, 2004

SoftVelocity Products Available Online At ClarionShop

ClarionShop is pleased to announce that for the first time ever, SoftVelocity products are available online. The IP Driver is $300, and the In-Memory

Driver is $225. Buy both together by Dec 31, 2004, and save $50.

Posted Wednesday, December 15, 2004

vuAgent And Valutilities Holiday Special

In the spirit of the holidays, Valutilities.com is extending a limited time promotion to all current registered customers. In addition to adding 44 new functions to vuFileTools (now a total of 116), all current registered owners of vuFileTools prior to (and shortly after) the release of version 3, will receive a free copy of Valutilities' newest product, vuAgent. vuAgent is an easy to use implementation of Microsoft's MS Agents, including Merlin, Peedy, Genie, and Robby, as well as a host of other free agents available over the internet. vuAgent also supports speech with the installation of Microsoft's free speech engine. vuAgent is as easy to use as calling vuAgentLaunch("Agent Name"). Want to make Merlin (or which ever agent you launch) do something like read, write, announce, or get your attention? Simply call vuAgentPlay("Play Command"). You can even send an entire sequence of commands to perform standardized actions. Current owners of vuFileTools will love vuAgent (especially the price)! All upgrade version 3 shipments of vuFileTools will include vuAgent, so registered customers need not do anything. Version 3 with vuAgent will be shipped to you automatically upon release.

Posted Wednesday, December 15, 2004

Jaguar Beta 3 Supports Cellphones/PDA Applications

JAGUAR v1.0 Beta 3 includes some minor improvements to the product, as well as many updates to the product documentation, but the main new feature is the ability to generate applications for mobile devices, such as MIDP-enabled cell phones, Palm devices, Pocket PC PDAs, etc. The JAGUAR Micro Edition, included in the Beta 3 of JAGUAR at no additional cost, is a separate template chain and a set of Java (J2ME) classes specifically designed for Clarion developers who want to design and generate applications for mobile devices. The templates included in the JAGUAR Micro Edition are the same that every Clarion programmer knows: Application, Window, Browse, Form, Source, etc. There are also wizards for rapid creation of applications for mobile devices.

Posted Wednesday, December 15, 2004

PostgreSQL Demo

Kristian Hyllestad has a PostgreSQL demo available for download. The program is a simple demonstration on a page loaded browsing using a DSN-less connection to a PostgreSQL database. The table used in demo contains approx. 285000 records containing company names and addresses. The program requires that you have the PostgreSQL ODBC driver installed on your machine. Tested with PostgreSQL 7.03. The installation is password protected; to get access, send an email to postgresql@spine.no.

Posted Wednesday, December 15, 2004

EasyListPrint 1.10

EasyListPrint version 1.10 is now available. Fixes include: Wrong blank DATE (@D) and TIME (@T) columns exported into Excel; Use of wrong separator for the decimals (now it comes from the OS system settings). This version for Clarion 5.0, 5.5 and 6.1 (9030).

Posted Wednesday, December 15, 2004

Arco Software Named Third-party Vendor of the Week

Hanspeter Stutz and Arco Software have been named by the Clarion Developers Challenge Football Contest as the Third-party Vendor of the Week! Hanspeter and Arco Sotware are awarding this week's winner of the Clarion Developers Challenge Football Contest a copy of their PiFolio Wordreporter 4 (Architect Version), the premier way to integrate MS Word report and document generation into your application.

Posted Wednesday, December 15, 2004

xTipHotKey Class v2.5

xTipHotKey Class v2.5 is now available. Changes include: FormatKeyCode method bug fixed; InteligentRemove method bug fixed; Compatible with Clarion 6.1 (build 9029).

Posted Wednesday, December 15, 2004

Clarion Developers Challenge Week 12 Winner

First place for Week 12 of the Clarion Developers Challenge goes to Stan Miller! Stan finished in a two-way tie with Al Henry in this week's contest, each having a total of 13 correct picks. Stan then pulled out the win in the Monday night tiebreaker by picking a tiebreaker score of 68, while Al picked a tiebreaker score of 49 (actual score was 62). All was not lost for Al, though, because with his great score he pulled into third place and one behind the leaders in the all-important season standings! For finishing first in this week's Clarion Developers Challenge, Stan will be receiving a copy of the Procedure Notes templates, donated by Eric Jacobowitz and Castle Computer Technologies, plus a copy of gReg that was re-donated to the Clarion Developers Challenge by last week's winner, Ed Schneider, and provided by Jesus Moreno and Gitano Software.

Posted Wednesday, December 15, 2004

Clarion Web Hosting

Clarion Web Hosting provides internet web for anyone who wants to use the Clarion Web Technologies. Contracts are a minimum of one month, from $10 a month.

Posted Wednesday, December 15, 2004

Quick Links

HOME

Subscribe/Renew

Clarion Books

FAQ (Frequently Asked Questions)

Log In

Download PDFs

Free Membership

Free Articles

Advertise

Reader Comments

Write For ClarionMag

Privacy

Contact