

For marketing your Applications
and Developer Accessories or to
purchase other 3rd Party Tools . . .

Developer tm **PLUS**

[Ads](#) · [Comments](#) · [Writers!](#) · [Privacy](#) · [Contacts](#) · [PDFs](#) · [Freebies](#) · [Open Source](#)

How To Ignore The Form Template

For the past few months, Steve Parker has been attempting to demonstrate that Clarion developers are often well-advised to ignore the templates that SoftVelocity provides. Having dealt with browses and reports, Steve now takes on the form template.

Posted Thursday, October 03, 2002

A Class For Printing Addresses

Printing street addresses on reports can be a pain - missing field data can make a good report ugly in a hurry. One way around this is to use a memo for addresses, but then you still have to format sometimes wildly different address data. Here's a small class that makes short work of address formatting.

Posted Thursday, October 03, 2002

How Not To Ignore The Form Template

In the previous episode, Steve Parker began "ignoring" the Form template, having previously ignored the report and browse templates. He stated that while he ignores the Form template's prompts the most, he ignores the Form template the least. In this final installment in the series, Steve explores what really matters about forms.

Posted Friday, October 04, 2002

PDF for September, 2002

All Clarion Magazine articles for September, 2002 in PDF format.

Posted Friday, October 04, 2002

Weekly PDF For October 1-5, 2002

All ClarionMag articles for October 1-5, 2002 in PDF format.

Posted Sunday, October 06, 2002

Review: Clarion/ASP, Part 4

In this fourth (and, he swears, final!) installment in his ongoing review of Clarion/ASP, Tom Hebenstreit looks at new features in 1.2 (and the soon-to-be-announced 1.3) and how to integrate Clarion/ASP with existing sites. And be sure to read Tom's summary of what Clarion/ASP is, and

SUBSCRIBE

CLARION

6 MONTHS
\$45

1 YEAR
\$80

2 YEARS
\$150

Subscribe Now

News

[Office Templates Gold Release](#)

[50% Off Gitano's gCal](#)

[EntabberPro Enter=Tab Solution](#)

[XScan 1.2 Released](#)

[SealSoft xFText 1.0](#)

[Browse Header Sort And Reverse Sort Template](#)

[xSmartMacro v2.2 Now Freeware](#)

[ExcelBond 2.0 Released](#)

[Gitano Software Affiliate Program](#)

[Clarion Third Party Profile Exchange Update](#)

[German Clarion Distributor Clearing Stock](#)

[EasyExcel And EasyReport Reviews At ClarionShop](#)

[INN Bio for 15-Oct-2002](#)

[New Version Of BoTpl](#)

[XP Menubar Version 3 Released](#)

who it's for.

Posted Tuesday, October 08, 2002

Remainder Of This Issue To Appear Next Week

This issue of Clarion Magazine will be split over two weeks: the remainder will appear early next week (which is a regularly scheduled off week).

Posted Friday, October 11, 2002

Not Having Six Makes Me Cranky

Why must SoftVelocity call the next release 5.6? Some say it's because at DevCon98 Bruce Barrington promised that version 6 would be 32-bit. Well, Carl Barnes was there, and he has a differing opinion.

Posted Tuesday, October 15, 2002

Clarion Challenge Results: ContainsMatch

We received a record number of entries for the Clarion Challenge to create a ContainsMatch function. Gordon Smith has analyzed and benchmarked the code, and provides illuminating commentary. And yes, we do have a winner.

Posted Thursday, October 17, 2002

Weekly PDF For October 6-19, 2002

All ClarionMag articles for October 6-19, 2002 in PDF format.

Posted Tuesday, October 22, 2002

Receiving Email With MAPI (Part 1)

MAPI email has been discussed in previous Clarion Magazine articles, but for the most part these articles cover sending email. In this two-part series, Konrad Byers shows how to receive email with MAPI.

Posted Wednesday, October 23, 2002

Data Structures And Algorithms Part X - Going Out Of Your Tree?

In the previous installment in this series, Alison Neal discussed the AVL Tree or Height Balanced Tree; in this article Alison continues her discussion of the AVL Tree and shows how to delete an individual item while still maintaining the sub tree heights.

Posted Thursday, October 24, 2002

ContainsMatch A to Z

In this followup to the recent Clarion Challenge, Phil Will examines what it would take to internationalize the ContainsMatch function.

Posted Thursday, October 24, 2002

[EasyMultiTag 2.03](#)

[CPCS Support delay](#)

[Replicate Sneak Preview](#)

[New Updates From Gitano Software](#)

[gSec At 50% Off](#)

[Free TPS Replication](#)

[ImageEx Review At ClarionShop](#)

[Credit Card Number Verification Algorithms](#)

[SimPad Templates](#)

[ThinkData Announces OutlookFUSE COM Automation For Microsoft Outlook](#)

[Firebird 1.5 Alpha](#)

[Simssoft Templates Update](#)

[Icetips Newsletter](#)

[New XPMenu Demo](#)

[INN Bio & News for 1-Oct-2002](#)

[IceTips Bio Notifications](#)

[Freeware SMTP MDA](#)

[PHP/SQL Info](#)

[Free Wallpaper](#)

[CapeSoft Incremental Upgrades](#)

[Upcoming CapeSoft Products](#)

[File Manager 3 In Beta Testing](#)

[Comm Ports, Excel Code, Etc.](#)

[Powerbrowse Site](#)

[Search the news archive](#)

Weekly PDF For October 20-26, 2002

All ClarionMag articles for October 20-26, 2002 in PDF format.

Posted Monday, October 28, 2002

Receiving Email With MAPI (Part 2)

MAPI email has been discussed in previous Clarion Magazine articles, but for the most part these articles cover sending email. In this series, Konrad Byers shows how to receive email with MAPI. Part 2 of 2.

Posted Wednesday, October 30, 2002

Converting The Inventory Example - Calling Stored Procedures (Part One)

In his previous articles, Ayo Ogundahunsi showed how to connect to SQL Server, import data, and generally run a Clarion application with a SQL Server backend. In this series of articles, Ayo shows how to call a stored procedure, pass a parameter, and (hopefully) get results from the stored procedure. Part one focuses on preparing the application by splitting it into DLLs.

Posted Thursday, October 31, 2002

Converting The Inventory Example - Calling Stored Procedures (Part Two)

In his previous articles, Ayo Ogundahunsi showed how to connect to SQL Server, import data, and generally run a Clarion application with a SQL Server backend. In this series of articles, Ayo shows how to call a stored procedure, pass a parameter, and (hopefully) get results from the stored procedure. Part two covers a template you can use to call stored procedures.

Posted Thursday, October 31, 2002

Looking for more? Check out the [site index](#), or [search the back issues](#).

This site now contains more than 700 articles and a total of over a million words of Clarion-related information.



[Ads](#) · [Comments](#) · [Writers!](#) · [Privacy](#) · [Contacts](#) · [PDFs](#) · [Freebies](#) · [Open Source](#)

[Topics](#) > [Using ABC](#) > [ABC, Using](#)

How To Ignore The Form Template

by **Steven Parker**

Published 2002-10-03

For the past few months, I have been attempting to demonstrate that Clarion developers are often well-advised to [ignore the templates](#) that SoftVelocity provides. Then I was asked "How does the form fit with your provocative assertion?" The "provocative assertion" being that a template only provides four things of use to a Clarion developer:

- a data section
- a screen/report formatter
- a View (where relevant)

and

- a processing loop

Because of these four facilities, I believe, it is not only possible but frequently desirable to ignore the template and call ABC methods directly in embedded source.

Of course, this is just an outrageous way of stating something that, on reflection, should be fairly obvious: the ABC templates are just wrappers for ABC methods. Just as the Clarion templates are wrappers for selectively calling Clarion language statements, the ABC templates selectively call ABC methods.

In past articles I've dealt with [reports](#) and [browses](#). With the form, my little crusade to provide a different perspective on template procedures nears its end.

The form, in fact, is the template whose *prompts* I ignore the most. This is largely because the Form template is the one that has changed the least in the last 10 years. In other words, I

continue using techniques that I've been using ... forever.

But, the Form template, as a whole, is the *template* I ignore the least. This template, perhaps because it is so well evolved, perhaps because what a form is expected to do is somewhat less involved than reports and browses, is in fact one that appears to add the least to the four critical features listed above.

The Prompts

In fact, the Form template provides only three prompts that are form-specific:

Field Priming on Insert

Allow: Inserts | Changes | Deletes

and

Messages and Titles

Except for the "Messages and Titles" prompt, I don't bother with the other two prompts very much. When I do use them, and I do, as much as anything, it is from a sense of obligation.

Let's take the nickel tour through the Form template prompts.

Field Priming on Insert

Field priming should come from the application's dictionary. Why, then, is there a template prompt to accomplish the same thing?

In fact, the "Field Priming on Insert" prompt generates code into `ThisWindow.PrimeFields`. In turn, `ThisWindow.PrimeFields` is just the locally derived method for affecting field priming defined in the dictionary.

This is easily verified by checking where the priming code is generated for fields you have set up in the dictionary, then priming an additional field in "Field Priming on Insert." Check the generated code and see where it ends up. Both your "manual" prime and the "automatic" priming should be generated together, in `ThisWindow.PrimeFields`.

You will, however, require access to this method. You do need to be able to do additional field priming; you need to be able to do field priming either omitted or inconvenient in the dictionary. Part of the reason is that you cannot use Clarion functions in the dictionary's "Initial Value" prompt or, at least, could not in earlier releases of Clarion. Consequently, I got

out of the habit of even trying to do so.

For instance, it is my recollection that earlier releases of Clarion threw compiler errors when using the `Today ()` function, for example, to prime a date field in the dictionary. Actually, when I tested this morning, in a Web Builder app, the field priming code compiled fine (so, it's no longer entirely true that you can't do this). However, the date field showed zero instead of the date.

I also had trouble priming a field in one file with a field from another in the Dictionary Editor.

If you need to prime a date field with `Today ()`, therefore, you may need some place to assign your date field. If you need to assign `PO:VendorCode = VND:VendorCode`, you need a place to make that assignment. Likewise, there may be assignments you forgot to make in the dictionary and you aren't motivated to update the dictionary. If you need to prime a variable in some procedures but not in others, you need to be able to do so on demand.

For over 10 years, I have been using the `Setup Procedure` embed in CDD to prime fields:

```
If LocalRequest = InsertRecord
  !field assignments
End
```

The `ABC ThisWindow.INIT` method (any embed after the files are opened) corresponds closely to CDD's `Setup Procedure` embed:

```
If ThisWindow.Request = InsertRecord
  !field assignments
End
```

I could (and sometimes do) use the `ThisWindow.PrimeFields` embed. This allows me to type only the assignments, omitting the `If ThisWindow.Request ... End` condition. However, this does not change the fact that this template prompt, while not just window-dressing, doesn't provide a lot that is new for me.

Allow: Inserts | Changes | Deletes

The series of three checkboxes for allowing inserts, changes, and deletes lets the developer restrict what an end user is allowed to do with a particular form.

My preference has always been to prevent the user from ever seeing the "You can't do this" message by not allowing them to press the button in the first place.

In CDD, if I disabled or deleted an action button, that action also became unavailable. Each button was logically separate from the other buttons in the set (the template checked for the

equate labels ?Insert, ?Change, ?Delete before including action code). A CDD button could be conditionally disabled in a browse's Process Selected Record embed. I like that. Because Clarion for Windows populates all three update buttons as a single Control template, it is a bit more work, but easy enough to replicate once you understand what is going on under the covers.

When a windows browse is initialized, properties controlling the buttons affecting insert, change and delete are set:

```
BRW1.Init PROCEDURE(SIGNED ListBox,*STRING Posit,VIEW V,QUEUE Q
? Start of "Browser Method Data Section"
? [Priority 5000]
? End of "Browser Method Data Section"
CODE
? Start of "Browser Method Code Section"
? [Priority 2500]
? Parent Call
PARENT.Init(ListBox,Posit,V,Q,RM,WM)
? [Priority 5500]
IF WM.Request <> ViewRecord
SELF.InsertControl=?Insert:3
SELF.ChangeControl=?Change:3
SELF.DeleteControl=?Delete:3
END
? [Priority 9300]
?right here
? End of "Browser Method Code Section"
```

Figure 1. Setting up browse buttons

In the embed tree, this code is found in the Browse on *<file Label>*, INIT, after Parent call embed (priority 9300). In that embed or at any time afterwards (e.g., ThisWindow.Init, 9001, for example), I can control what buttons are available to the user:

```
If OPT:Scenario <> 'S' and OPT:Scenario <> 'H'
BRW1.DeleteControl = 0
Disable(?Delete:3)
BRW1.InsertControl = 0
Disable(?Insert:3)
Disable(?CopyButton)
Disable(?ChangeIDButton)
End
```

The code above prevents users from either deleting or inserting records *and* disables the buttons. No messages, no pop-up menus, no needless button presses ... *my style*.

And, note that what I have done here is conditional. The controls are only turned off on clearly testable conditions. The template prompt does not give me that flexibility (nor would I, reasonably, expect it to) and the template, affecting the form, does not disable the browse's buttons.

I can also use the `NewSelection` method to turn buttons on and off while scrolling through a list box, as shown in Figure 2:

```
BRW1.TakeNewSelection PROCEDURE
? Start of "Browser Method Data Section"
? [Priority 5000]

? End of "Browser Method Data Section"
CODE
? Start of "Browser Method Code Section"
? [Priority 2500]

? Parent Call
PARENT.TakeNewSelection
? [Priority 5001]
| If UND:PollingName and POH:POStatus = 'Open'
  Enable(?Transmit)
Else
  Disable(?Transmit)
End
If POH:POStatus = 'Sent'
  Disable(?Change:3)
End
? End of "Browser Method Code Section"
```

Figure 2. NewSelection controls buttons while scrolling

Looking at Figure 2, I'm glad I chose this example. There's an error in the code and I now get to add:

```
Else
  Enable(?Change:3)
End
```

to the `If POH:POStatus` condition. This is needed to ensure that the button is properly re-enabled after passing through a record that disabled it.

The `Allow: Inserts | Changes | Deletes` template option does provide some useful functionality, but nothing I can't do and do more to my own tastes in less than a minute. The template, however, is very useful in showing me how to do what I want to do.

Messages and Titles

Finally, here's a Form template prompt I use, and use with regularity.

This prompt lets me select the message the users see when they cancel a form ("On Aborted Add/Change"), whether a message is displayed at all, and how deletes are handled ("Standard Warning," "Display Form," "Automatic Delete"). This template also allows configuration of recursive adds and action messages (note that I turn them off in "Location of Message").

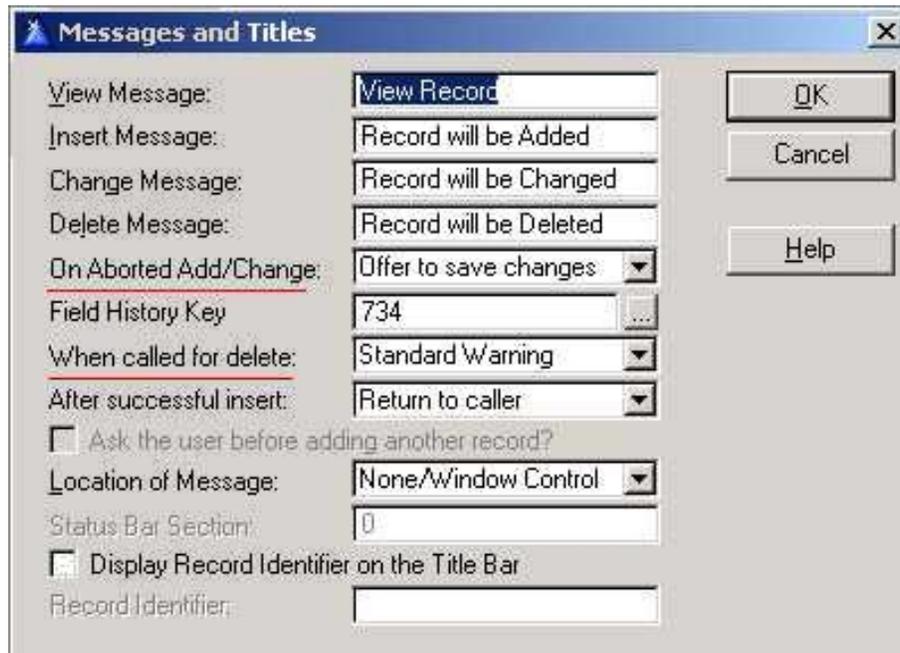


Figure 3. Form prompts

I use these prompts but not because this template prompt offers special access to these properties. They are easy to use and, by not having to trace `Self.InsertAction`, `Self.CancelAction` and `Self.DeleteAction`, the ABC properties set by these prompts, the template allows me to set-and-forget them. Set-and-forget is what I usually want.

If I found myself needing to reset any of these at runtime, I'd ignore these prompts too. I'd start investigating the relevant properties and setting the properties at the applicable embed(s).

For example: suppose I wanted a browse where I could call the update form for either a regular (single) add or for recursive adds. Suppose I want to supplement the standard Insert button with a "batch" button.

I know that the template supports this. Note the "After successful insert," prompt in Figure 3, above. (Here's how you find out what properties are being affected: open the Form template – from the Clarion main menu, Setup | Template Registry – and search for the prompt text.)

If I look at the code generated when "Return to caller" is selected, I see Figure 4:

```

? Open Files
Relate:ABINUTRY.Open
Relate:ABPODETL.Open
SELF.FilesOpened = True
SELF.Primary &= Relate:ABPOHEAD
IF SELF.Request = ViewRecord
  SELF.InsertAction = Insert:None
  SELF.DeleteAction = Delete:None
  SELF.ChangeAction = 0
  SELF.CancelAction = Cancel:Cancel
  SELF.OkControl = 0
ELSE
  SELF.CancelAction = Cancel:Cancel
  SELF.OkControl = ?OK
  IF SELF.PrimeUpdate() THEN RETURN Level:Notify.
END
? [Priority 7501]

```

Figure 4. Code generated for "Return to caller"

If I now select "Insert another record," I see the code in Figure 5:

```

? Open Files
Relate:ABINUTRY.Open
Relate:ABPODETL.Open
SELF.FilesOpened = True
SELF.Primary &= Relate:ABPOHEAD
IF SELF.Request = ViewRecord
  SELF.InsertAction = Insert:None
  SELF.DeleteAction = Delete:None
  SELF.ChangeAction = 0
  SELF.CancelAction = Cancel:Cancel
  SELF.OkControl = 0
ELSE
  SELF.InsertAction = Insert:Batch
  SELF.CancelAction = Cancel:Cancel
  SELF.OkControl = ?OK
  IF SELF.PrimeUpdate() THEN RETURN Level:Notify.
END
? [Priority 7501]

```

Figure 5. Code generated for "Insert another record"

Comparing the two, I see how it's done:

<pre> ! Open Files Relate:ABINUTRY.Open Relate:ABPODETL.Open SELF.FilesOpened = True SELF.Primary &- Relate:ABPOHEAD IF SELF.Request = ViewRecord SELF.InsertAction = Insert:None SELF.DeleteAction = Delete:None SELF.ChangeAction = 0 SELF.CancelAction = Cancel:Cancel SELF.OKControl = 0 ELSE SELF.CancelAction = Cancel:Cancel SELF.OKControl = ?OK IF SELF.PrimeUpdate() THEN RETURN Level:Notify. END ! [Priority 7501] </pre>	<pre> ! Open Files Relate:ABINUTRY.Open Relate:ABPODETL.Open SELF.FilesOpened = True SELF.Primary &- Relate:ABPOHEAD IF SELF.Request = ViewRecord SELF.InsertAction = Insert:None SELF.DeleteAction = Delete:None SELF.ChangeAction = 0 SELF.CancelAction = Cancel:Cancel SELF.OKControl = 0 ELSE SELF.InsertAction = Insert:Batch SELF.CancelAction = Cancel:Cancel SELF.OKControl = ?OK IF SELF.PrimeUpdate() THEN RETURN Level:Notify. END ! [Priority 7501] </pre>
---	--

Figure 6. Differences in generated code

This suggests that I can, at run time, change `SELF.InsertAction` from nothing (property not assigned, as in Figure 4) to `Insert:Batch` (as in Figure 5).

The sample app which accompanies this article demonstrates that this is precisely the case. Though, because the template prompts for `Update Procedure` do not yet take parameters, I am "reduced" to using a global variable to indicate whether to batch add or not.

Using a global variable is not as elegant as one might wish. But it does have the virtue of working. That's a tough virtue to beat.

Obviously, I could do the same for the rest of the prompts/properties on this sheet, *if* I found myself needing to reset any of them at runtime.

Summary

Okay. If the Form template is so ... "uninteresting," I guess is the word, why are forms often so difficult?

The answer, of course, is as I stated at the outset: the Form template "adds the least to the four critical features listed." The most critical of these feature, at least as far as a form is concerned, is the processing loop.

I'll examine *that* [next](#).

[Download the source](#)

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

Reader Comments

[Add a comment](#)

Field Priming on Insert I know for sure in Local, Module...

Copyright © 1999-2002 by [CoveComm Inc.](#) All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited.

INVEST
in your own abilities**Clarion**
magazine[Ads](#) · [Comments](#) · [Writers!](#) · [Privacy](#) · [Contacts](#) · [PDFs](#) · [Freebies](#) · [Open Source](#)[Topics](#) > [Reports/Processes](#) > [Reports and Processes](#)

A Class For Printing Addresses

by **David Harms**

Published 2002-10-03

I've always found street addresses a bit of a pain in reports. The difficulty is that addresses vary so widely, whether between countries or between counties. One address will have a company name, the other just an individual name. Perhaps the individual is listed by first and last name, or last name only. And you may or may not have postal codes for all addresses, and so forth.

I long ago abandoned using individual string fields for report addresses, and instead populated a memo (a technique also [described by Steve Parker](#)), to which I added each address, line by line. But of course that meant I had to copy and paste that code whenever I created a new report, or write a function to receive all possible address fields as parameters, and massage these into a proper address. And not long after Clarion sprouted OO capabilities, I realized that this particular problem was an ideal application for a class.

Using the `cciAddressClass`

The `cciAddressClass`, which I'll describe in a moment, is designed to make it relatively painless to format street addresses for reports. To use it, you'll need to do the following:

1. Populate a memo field on the report for the address
2. In a global or module data embed, add the following:

```
include('cciaddr.inc')
```

Although `cciAddressClass` is recognized by the IDE as an ABC class, it doesn't (yet) have a template which makes it known to the IDE. If you don't specify the include statement, you'll

get a bunch of Illegal data type, Field not found, and Unknown procedure label syntax errors.

3. Declare an instance of cciAddressClass in your procedure's data section:

```
AddressObj cciAddressClass
```

4. For each data record, load AddressObj with your data and then retrieve the resulting address into the report's address memo

Here's an example:

```
AddressObj.Reset()
AddressObj.AddLine(nam:FirstName,nam:LastName)
AddressObj.AddLine(nam:Company)
AddressObj.AddLine(nam:Address1)
AddressObj.AddLine(nam:Address2)
AddressObj.AddLine(nam:City,',',nam:StateProv,nam:Postal)
Address = AddressObj.getAddress()
```

The first call is to the Reset() method, which clears any previously set data. The next five calls are all to the AddLine method, which loads up the address data, and the last call retrieves a string, complete with carriage returns/line feeds, suitable for printing as a memo. Here's the source for the class methods:

```
cciAddressClass.AddLine    procedure(string Part1,<string Part2> ,|
                           <string Part3>,<string
Part4>,<string Part5>)
temp      cstring(500)
temp2     cstring(500)
x         long
code
temp = clip(Part1)
if ~omitted(3) then self.AppendToLine(temp,part2) .
if ~omitted(4) then self.AppendToLine(temp,part3) .
if ~omitted(5) then self.AppendToLine(temp,part4) .
if ~omitted(6) then self.AppendToLine(temp,part5) .
temp = left(temp)
! strip off any leading commas
loop x = 1 to len(temp)
  if temp[x] = ',' then temp[x] = ' ' else break.
end
! strip off any trailing commas
loop x = len(temp) to 1 by -1
  if temp[x] = ',' or temp[x] = ' '
    temp[x] = ' '
  else break.
end
self.AddOneLine(left(temp))

cciAddressClass.AddOneLine procedure(string Line)
code
```

```

    if len(clip(line)) > 0
        self.AddressQ.Line = Line
        add(self.AddressQ)
    end

cciAddressClass.AppendToLine      procedure(*cstring cs,string s)
    code
    ! If s is or begins with a comma, append it
    ! directly, otherwise add a space.
    if s[1] = ','
        cs = cs & clip(left(s))
    else
        cs = cs & ' ' & clip(left(s))
    end

cciAddressClass.Construct        procedure
    code
    self.AddressQ &= new(AddressQueue)

cciAddressClass.Destruct        procedure
    code
    self.Reset()
    dispose(self.AddressQ)

cciAddressClass.GetAddress      procedure
temp      string(500)
x         long
    code
    loop x = 1 to records(self.AddressQ)
        get(self.AddressQ,x)
        if x = 1
            temp = self.AddressQ.Line
        else
            temp = clip(temp) & '<13,10>' & self.AddressQ.Line
        end
    end
    end
    !stop('GetAddress ' & temp)
    return(temp)

cciAddressClass.Reset          procedure
    code
    free(self.AddressQ)

```

For the most part, the source is straightforward. Each time you pass the class some data, it stores it in a queue, unless the string is empty. That takes care of blank lines. The rest of the code attempts to deal with commas and missing address fields. If, for instance, you are expecting data as

```
city, state zip
```

And the city has been entered incorrectly in, say, a second address field, your address could come out as

```
city  
,state zip
```

To correct this, the code strips off leading commas, which can be a big help with malformed addresses. As well, the AppendLine method (which is private to the class) automatically adds a space to the line if there's already some text present. The class also removes trailing commas, which is useful if you are listing names as `lastname , firstname` and there is no first name.

No doubt there are many possible improvements and enhancements to this class; feel free to post these as reader comments. You may also want to read two other Clarion Magazine articles: Steve Parker's [Standard Address Handling](#), which I mentioned earlier, and Carl Barnes' [Using CHOOSE\(\) To Concatenate Data](#)

[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). His most recent book is [JSP, Servlets, and MySQL](#), published by HungryMinds Inc. (2001).*

Reader Comments

[Add a comment](#)

There sure are alot of good ways to do this! I put an...

Watch out that fame does not go to your head - you used to...

Tony, Thanks! ** Dave **

Copyright © 1999-2002 by [CoveComm Inc.](#) All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited.

Reborn Free

CLARION
online

[Ads](#) · [Comments](#) · [Writers!](#) · [Privacy](#) · [Contacts](#) · [PDFs](#) · [Freebies](#) · [Open Source](#)

[Topics](#) > [Forms](#) > [Forms - using](#)

How Not To Ignore The Form Template

by **Steven Parker**

Published 2002-10-04

For the past few months, I have been attempting to demonstrate that Clarion developers are often well-advised to ignore the templates that SoftVelocity provides. I realize, of course, that it is not entirely easy to ignore the very device you are using to create your procedures and applications.

Oddly enough, however, I think most Clarion developers actually use the approach I have been advocating. Now, I hope, more of you will be fully cognizant of the fact that you, too, are template ignorers at the very same time you are template users (consistency just doesn't appear to be one of our problems).

What I want to emphasize is that Clarion developers need to understand that the templates are just wrappers for the ABC classes. Therefore, Clarion developers use this fact and, with at least a basic understanding of default template behavior, simply call the desired methods directly, at the time and place of their choosing.

In the [previous episode](#), I began ignoring the Form template (I have already ignored the [report](#) and [browse](#) templates). I stated that while I ignored the Form template's *prompts* the most, I ignored the Form *template* the least.

The Form is *my* least ignored template because it is the one, other than the Process template, that has the least effect on how the core ABC classes do their job. In ignoring browses and reports, I had to learned about a number of ABC methods and about using those methods within the templates' embeds. But, there isn't much to learn from the Form template in this regard. This is precisely because form handling revolves exclusively and obviously around the ACCEPT loop and, while report and process handling have changed a lot with ABC, the

ACCEPT loop remains much as it has been for years.

An understanding of what happens inside a Form's ACCEPT loop becomes important because such template features as do exist are, in effect, magnified by the essential simplicity of the Form.

So, instead of attempting to ignore the Form, I want to look at ACCEPT processing in this article.

Form Basics

Just as a report is a "[PRINT\(\) in a loop](#)," a Form is nothing but a "Case in a Loop." It's just that simple.

Right.

In it's barest form, the logic of a Form looks like this:

```
Accept
  Case Event()
  Of <whatHappened?>
    <whatToDoAboutIt>
  End
End
```

In a somewhat more useful rendition (and one that is pretty close to what the Clarion Legacy templates generate), it looks more like this:

```
Accept
  Case Event()
  Of Event:OpenWindow
  Of Event:GainFocus
  !other "window" events
  Of Event:Selected()
    Case Field()
    Of ?Field1
    Of ?Field2
    !etc
  End
End
Of Event:Accepted()
  Case Field()
  Of ?Field1
  Of ?Field2
  !etc
  End
End
End
```

Processing nested CASEs ensures that all embeds containing code get executed at (more or less) the correct time. If you take a minute to think through the code schematic above, you will see that whatever event occurs *will* cause the correct embeds to execute. You should also note that only one event can be handled at a time (stay tuned, this might be important). Naturally, events are also handled in the order in which they occur.

In the ABC classes, the TakeEvent method is the master method for the Accept loop. TakeEvent calls TakeWindowEvent, which handles events like GainFocus, LoseFocus, OpenWindow, CloseWindow, ShutDown and the like. TakeEvent also calls TakeSelected, which handles the field Selected embeds, TakeAccepted, for handling Accepted embeds, and a few other methods.

Take a look at the TakeEvent method code (found in abwindow.clw):

```
WindowManager.TakeEvent PROCEDURE
RVal BYTE(Level:Benign)
I    USHORT,AUTO
CODE
  IF ~FIELD()
    RVal = SELF.TakeWindowEvent()
    IF RVal THEN RETURN RVal.
  END
CASE EVENT()
OF EVENT:Accepted
  RVal = SELF.TakeAccepted()
OF EVENT:Rejected
  RVal = SELF.TakeRejected()
OF EVENT:Selected
  RVal = SELF.TakeSelected()
OF EVENT:NewSelection
  RVal = SELF.TakeNewSelection()
OF EVENT:AlertKey
  IF SELF.HistoryKey AND KEYCODE() = SELF.HistoryKey
    SELF.RestoreField(FOCUS())
  END
END
IF RVal THEN RETURN RVal.
IF FIELD()
  RVal = SELF.TakeFieldEvent()
  IF RVal THEN RETURN RVal.
END
LOOP I = 1 TO RECORDS(SELF.CL)
  GET(SELF.CL,I)
  RVal = SELF.CL.WC.TakeEvent()
  IF RVal THEN RETURN RVal.
END
IF SELF.AutoRefresh
  LOOP I = 1 TO RECORDS(SELF.CL)
    GET(SELF.CL,I)
    IF SELF.CL.WC.ResetRequired()
      SELF.Reset
```

```

        BREAK          ! One reset per cycle is quite enough...
    END
END
END
END
RETURN Rval

```

Note how `TakeEvent` starts by checking for a field-independent (i.e., window) event via the `If ~FIELD()` test. From the online Help:

The `FIELD` procedure returns the field number of the control which has focus at the time of any field-specific event. This includes both the `EVENT:Selected` and `EVENT:Accepted` events. `FIELD` returns zero (0) for field-independent events.

This means that when `FIELD()` is zero (i.e., `~FIELD()`), a window event, not a field event, occurred.

If a field-specific event did occur, `CASE EVENT()` is evaluated. This is where `Selected` and `Accepted` embeds are executed, if there is any code to execute. The corresponding methods are, of course, `TakeSelected` and `TakeAccepted`.

If none of the methods in the `CASE EVENT()` structure returns a value, `TakeFieldEvent()` is called. (Note: `TakeEvent` returns on the *first* successful call to another method.) `TakeFieldEvent()` is a method I can't claim much familiarity with, but I discovered that it is called from the `All Events` embed point.

I had never used the `All Events` embed before I looked at the `TakeEvent` method code for this article. As the downloadable demo app shows, `TakeFieldEvent` is called whenever an event is posted for the control; any event, all events. In the demo app I put a message in a couple of `All Events` embeds and found that `TakeFieldEvent` is indeed called when the control is selected or when it is completed.

I am not entirely sure why the `All Events` embed/`TakeFieldEvent` method was created. I always use the controls' `Accepted` and `Selected` embeds/methods. I suppose that if you had a user defined event that needed to be acted on by a specific control, you would embed your code here. In any case, it's useful to know it's there.

RTL Feature #1

The `ACCEPT` loop will cycle for every event that might need to be handled. Therefore, if there is code that needs to execute for a particular event, you can be sure it will.

When a user tabs off a field, for example, an `Event:Accepted` may occur for that field, and

an `Event : Selected` will occur for the field or button "tabbed onto." Timer events, window events, etc. all cause the `ACCEPT` loop to cycle. As Richard Taylor explained in a newsgroup posting:

The `ACCEPT` loop cycles once for every event that occurs in your procedure that your code might need to handle. Some actions fire multiple events. For example: opening the window fires `EVENT : OpenWindow` and `EVENT : GainFocus`; tabbing from one control to another fires `EVENT : Accepted` for the control you're moving from (if the user entered data) then `EVENT : Selected` for the control you're moving to.

If the `ACCEPT` loop appears to cycle multiple times, this is why: multiple Windows events can be posted by a single user action. (See? I told you the handling of one event at a time might be important.) `ACCEPT` cycles for *each* event; tabbing or clicking, for example, can involve two events (an `EVENT : Accepted` and an `EVENT : Selected`).

If you have embedded code in one of the methods affected by this `CASE`, it will execute.

The apparent exception to this is `Event : Accepted`. Note that I stated that "an `Event : Accepted` *may* occur." And Richard Taylor stated "*if* the user entered data." This implies that if there is code in a field's `Accepted` embed but the user simply tabs through the field, any `Event : Accepted` code will not execute.

Not only is this exactly what happens, it is designed in. This is what is intended by Clarion's designers (in other words, this is not ABC specific). The Language Reference Manual states:

EVENT:Accepted

The user has entered data or made a selection [in a drop down or option control, for example] then pressed `TAB` or `CLICKED` the mouse to move on to another control. This is the event on which you should perform any data input validation code.

Not firing `Event : Accepted` when simply passing through a field reduces overhead by eliminating a complete cycle of `ACCEPT`. But you may *want* the embedded code to execute whether or not the user did anything in the field.

Understanding what the RTL is doing (and this is a feature of the RTL, not the template), in not executing `Event : Accepted` when the user has done nothing in the field, gives the key to bypassing this built-in behavior.

To force code to execute, embed the code:

```
?pre:field{Prop:Touched} = True
```

in that field's Selected embed. Here's what the Help has to say about PROP:Touched:

PROP:Touched

When non-zero, indicates the data in the ENTRY, TEXT, SPIN, or COMBO control with input focus has been changed by the user since the last EVENT:Accepted. This is automatically reset to zero each time the control generates an EVENT:Accepted. Setting this property (in EVENT:Selected) allows you to ensure that EVENT:Accepted generates to force data validation code to execute, overriding Windows' standard behavior--simply pressing TAB to navigate to another control does not automatically generate EVENT:Accepted.

Feature #2: Data Validation

Placing data entry fields inside a looping structure like Clarion's ACCEPT statement, is ready made for validating a datum against another file.

Because of this, the Must Be In File restriction in the Dictionary Editor is simply unnecessary. It is most convenient that forms are structured this way because "Must Be In File" is notorious for causing trouble. (If you have a "Must Be In File" restriction in your dictionary and, for example, remove one of the files or change key definitions, the templates will still generate the code for "Must Be In File." But, of course, the file is gone and the app will not compile.)

Because an event is generated when a field is selected, it is exceedingly easy to force a user to select a value. Because another event is generated when a field is changed (or, with PROP:Touched, thinks it has), it is exceedingly easy to verify that the value exists in another file:

```
LookupFileKeyValue = pre:field
Get(LookupField,LookupKey)
If ErrorCode() !call a lookup procedure
```

And this code will work in either the Accepted or Selected embed.

In fact, ACCEPT as implemented in a Form is so validation-friendly that code templates to affect this are pre-populated in the Form Template. These code templates take care of generating code like that shown above for you.

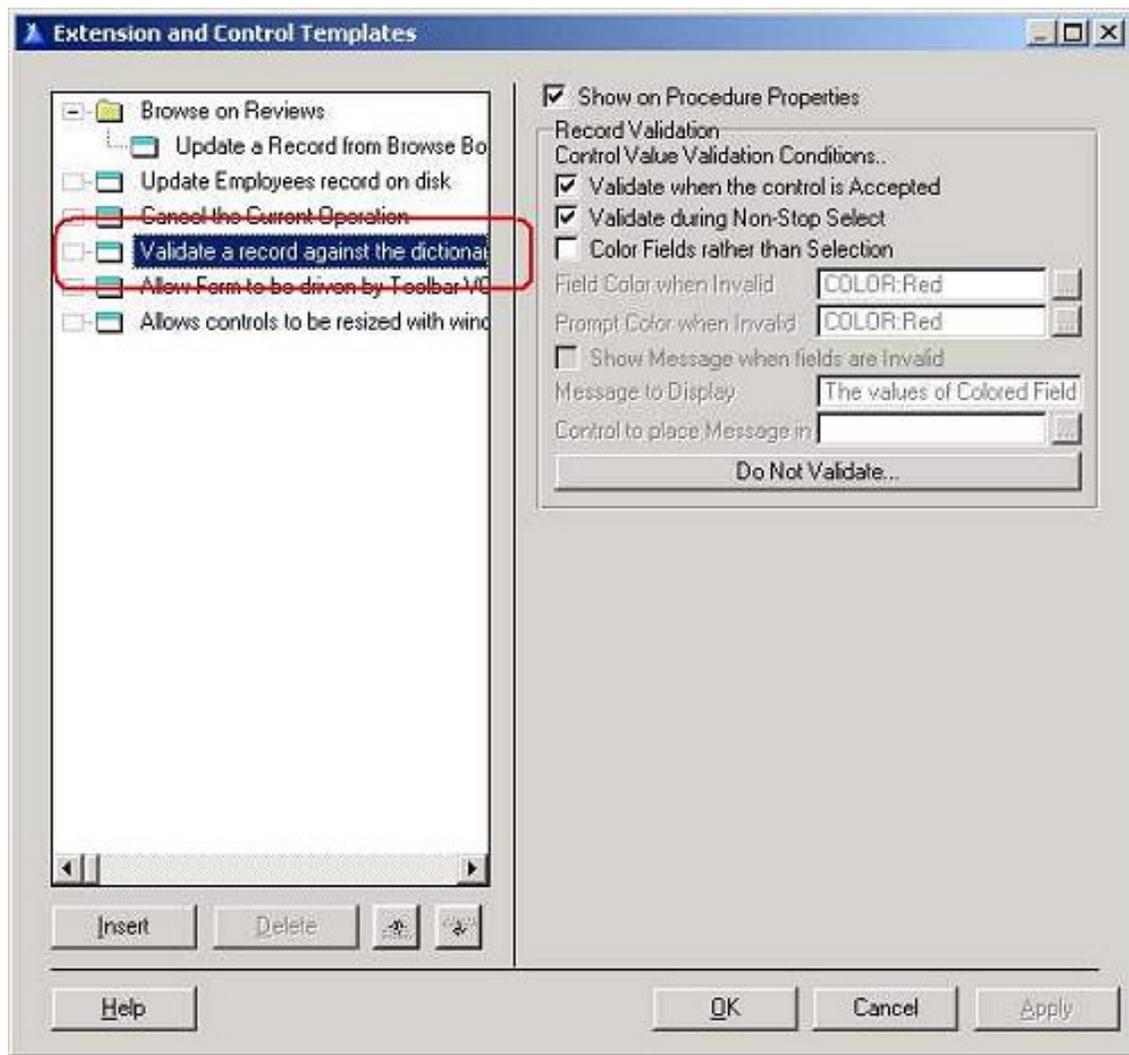


Figure 1. Validation extension pre-populated

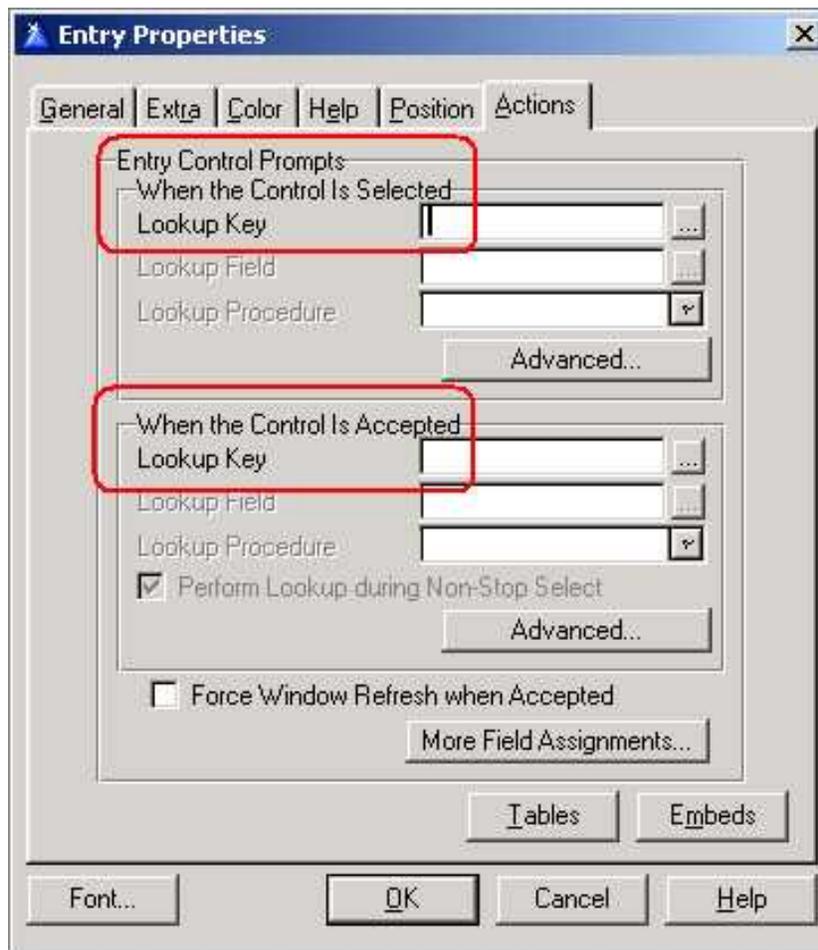


Figure 2. Built-in field lookup options

For more customized lookups, there is the `CallProcedureAsLookup` Code template that you can populate virtually anywhere. This template provides additional embeds that enhance lookup flexibility.

Of course, when all else fails, the `Selected` and `Accepted` embeds allow full hand coding of lookups.

All of these validation techniques have been discussed before. Check out, for example, [Lookups in C4](#) in the Clarion Online archives (this article was published in 1998 and is still entirely applicable) or [Sometimes Lookups](#), [Lookups – You Don’t Always Want to Validate](#), among other articles.

This feature, a built-in validating structure, is not provided by the RTL but neither, I think, is it fair to call it a template feature. It is a very happy consequence of the structure of Clarion Forms.

Feature #3: Data Validation, Required Fields

Required field checking is not, strictly speaking a feature of the template. Instead, the RTL

uses the looping nature of a Clarion Form to check and act on the REQ attribute.

In many, if not most cases, a field is marked "Cannot Be Zero or Blank" in the dictionary. This causes the field to be populated with the REQ attribute (this attribute can be added to a field on a window even if it is not checked in the dictionary).

If a button (usually the Ok button pre-populated on a form or the SaveButton template which you can populate on a plain window) also has the REQ attribute, as it does on a default Form, required fields will be checked before anything else happens when completing the form. (Yup, "REQ" looks like one of those overloaded thingies.)

The REQ attribute (PROP:REQ) specifies an ENTRY or TEXT control that may not be left blank or zero. The REQ attribute on an ENTRY or TEXT control is not checked until a BUTTON with the REQ attribute is pressed, or the INCOMPLETE () procedure is called.

When a BUTTON with the REQ attribute is pressed, or the INCOMPLETE () procedure is called, all ENTRY and TEXT controls with the REQ attribute are checked to ensure they contain data. The first control encountered in this check that does not contain data immediately receives input focus.

It is also possible to use the Incomplete () procedure to manually check required fields (without the REQ attribute on the button). Incomplete () has the virtue of making it easy to add a user friendly message when a field is left empty. See the series of articles on required fields starting with [First Field, Required Field](#) .

Feature #4: The Double Loop

Oh dear, this one is fun.

An Ok button pre-populated on a Form (or the SaveButton template on plain windows) adds a little something extra to the Form. And what it adds is extremely important, the "Double Loop."

There is a problem with Forms in Windows and that problem is that, while the window may open with the first field selected, the user can move to any other field on the form and can do so at any time. Consider a rudimentary retail store inventory entry form:

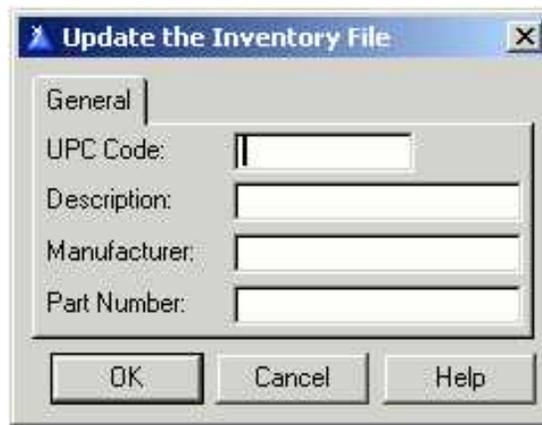


Figure 3. Basic inventory form

The UPC Code would typically be a required field. But a user could arbitrarily click on or tab to Description, make an entry and press "Ok" (and, no, I am not making this one up).

If this form could be saved with UPC Code blank, just what is the point of having marked the field "Required?" How is required field checking carried out?

What the Form's Ok button does is to cause the templates to add the `Select` statement (notice the absence of parameters) to the button's actions (see [A Closer Look at Required Fields](#) for a fuller discussion).

The on-line help states "SELECT with no parameters initiates `AcceptAll` mode (also called non-stop mode)."

[`AcceptAll`] is a field edit mode in which each control in the window is processed by generating `EVENT:Accepted` for each. This allows data entry validation code to execute for all controls, including those that the user has not touched. `AcceptAll` mode immediately terminates when any of the following conditions is met:

`Select(?)`

`Window{PROP:AcceptAll} = 0`

A `REQ` control is left blank or zero.

Read that again because *that* is how required field checking is enforced. It is also the most import feature of Forms with which one *must* come to grips because it also tells you how to modify the standard behavior.

This means that every `Accepted` embed is executed when the window is in non-stop mode,

including any embeds already executed. This explains why embedded code may execute twice (interesting, isn't it, that a few paragraphs back I was commenting on how code in an `Accepted` embed might not execute at all and, here, I notice it can execute twice).

If I embed:

```
INV:UnitPrice = (INV:Cost * INV:Markup) / INV:PackQty
```

multiple execution doesn't really matter; the data isn't going to be changed in unexpected ways.

But suppose I want to compute costs only once because recalculating *can* overwrite a manual change. Or suppose I increment a value or counter: it would increment twice or it could increment simply by looking at the form and pressing Ok. In these cases, knowing about and checking for non-stop mode allows me to short circuit the embed and prevent multiple execution:

```
If ~0{Prop:AcceptAll}
  DO CalculateCost
End
or
If ~FormWindow{Prop:AcceptAll}
  DO RoundPrice
End
```

In this way, I ensure that the calculations are made only when the field has actually been changed and, in fact, only immediately after that change is made (conveniently making it quite easy for me to display these calculations if I need to).

Developers frequently want to intercept the "Creates Duplicate Key" message when they would rather users not see it (and the developer does not want to constantly [modify the standard error messages](#)). Usually, this is done by checking the `DUPLICATE` function, in the field's `Accepted` embed:

```
If ThisWindow.Request = InsertRecord
  Get(Inventory,0)
  If Duplicate(INV:UPCKey)
    Message('This Code already exists in the inventory.'|
      , 'Duplicate UPC',ICON:Hand)
    Select(?)
    Cycle
  End
End
```

However, when the form is completed and goes into non-stop mode, `DUPLICATE` will return `True` even though the value *is* unique (try it, you won't like it). Changing the code to:

```

If ThisWindow.Request = InsertRecord |
  and ~0{Prop:AcceptAll}
Get(Inventory,0)
If Duplicate(INV:UPCKey)
  Message('This UPC already exists in the inventory.' |
    , 'Duplicate UPC', ICON:Hand)
  Select(?)
  Cycle
End
End

```

prevents the check from happening a second time.

If you've been around Clarion for a while, you're used to the double loop. If you're a recent convert, the double loop may seem like a questionable design decision.

Can you think of a better way, a way that can be built into a template, for enforcing required fields? Besides, it can be easily by-passed.

Feature #5: Completing a Form

A lot of folks need to do "post-processing" on a data entry form. Often, too often, this was done in `Ok Accepted, Before Parent call`. But, Windows, don't you know, has dictated that life be more complex. If you have selected any kind of message or confirmation on a Cancel button press, as in Figure 4, the user will get another chance to save the record.



Figure 4. Standard Windows Warnings

And, if the user does save the record, the code in `Ok Accepted, Before Parent`

call may not execute (you may already logically be past that point in the code).

Move all such code to the `TakeCompleted` method (before parent call). This method is always called whether the Ok button is pressed or the user cancels and reconsiders.

Summary

I was asked, "How does the form fit with your provocative assertion [that there are only four things important in a template: a data section, a window/report formatter, a View and a processing loop]?" The answer, for once, is that I don't ignore the Form template. I don't because the Form fits quite nicely with my preconceptions about what is important in template procedures. But, understanding how the Form template operates *is* important.

I don't ignore the Form template. But, as in the case of the double loop for example, I do find that I sometimes need to ... by-pass it. The power of the Form template, then, is not in the specific methods it calls but in understanding the `ACCEPT` loop and how the Form template depends on it.

[Download the source](#)

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

Reader Comments

[Add a comment](#)

It has been pointed out that issues concerning the OK...

Reborn Free

CLARION
online

[Ads](#) · [Comments](#) · [Writers!](#) · [Privacy](#) · [Contacts](#) · [PDFs](#) · [Freebies](#) · [Open Source](#)

[Topics](#) > [Internet](#) > [ASP](#)

Review: Clarion/ASP, Part 4

by Tom Hebenstreit

Published 2002-10-08

Welcome to the final installment of the Clarion Magazine [Clarion/ASP](#) review. If you haven't read the [first](#), [second](#) and [third](#) installments yet, I recommend you go through them first to get the broad picture of how Clarion/ASP works.

Before I jump into the final sections on embeds and integration, let's get caught up on the latest news for Clarion/ASP.

Clarion/ASP v1.2 Released

In mid-September, SoftVelocity released a second update to Clarion/ASP to bring it up to version 1.2. New features and enhancements included (as listed in the v1.2 ReadMe file):

- **New feature:** The runtime HTML template used for a given procedure may be substituted via a URL parameter of "?HTMLT=someurl.htm" or by setting the VBScript variable `HTML_Template` to some value at the `BeforeMergeCall` embed. Note that the location must be relative to the ASP script directory or absolute with a leading `"/`". This feature is implemented for form and browse procedures; not message (status) boxes.
- **New feature** - Addition of German language translation file.
- **New feature** - Added Global option to specify the ADO command type on `recordset.open` methods for List, Edit, Add and View pages.
- **New feature:** Added support on search forms to include lookup buttons for fields when appropriate. Uses the same logic as Add or Edit forms.
- **New feature** - Date lookup routine now supports using the current month/year if entered.
- **Change in behavior** - Removal of the Help pop-up resize function which ignored the template settings for pop-up size.

- New feature - New option for HTML List procedures to flag the generation of blank "padding" rows.
- Enhancement - Additional logic added to better support data columns represented as check boxes.
- Enhancement - Column names with embedded spaces are now supported.
- Change in behavior - Date column test included instring search for "DATE"; removed.

A number of bugs were also fixed in this release. On a personal level, I'm pleased to report that SoftVelocity seems to be listening closely to suggestions for improvements to Clarion/ASP (including those made here by yours truly.) After grouching in the previous installment about the Regenerate SQL button not being on the SQL tab in the templates, guess where it is in v1.2? That's right, it's now right there on the SQL tab along with the SQL code that it affects. Thanks, guys!

Installing the Clarion/ASP v1.1 to 1.2 patch

The update for v1.2 was made available as a downloadable patch from the private Clarion/ASP newsgroups. Running it updated the templates and online help file to the new version. The two PDF manuals for Clarion/ASP were also updated for v1.2, but have to be downloaded separately from the SoftVelocity web site. Why they didn't just include a direct download link right at the v1.2 download message, I don't know. I do know that it was a needlessly complex process to obtain the revised manuals, since at the time of this article (about three weeks after the v1.2 update had been released), the SoftVelocity web site has not been fully updated to reflect the v1.2 release.

For example, although the SoftVelocity home page mentions the v1.2 update, following the links to new Clarion/ASP features displays a page on the v1.1 update. Finding the new manuals I was seeking was also problematic, as they were hidden away under the Product Information link. Once I finally found them, there was nothing at all on the Product Information download page to indicate that the manuals had been updated in any way since the original release of Clarion/ASP. It would be very helpful for SoftVelocity to post the version and/or date of the manuals on the download page along with the filename so that you could easily tell if a download was needed or not.

Final notes on the update: If you have been modifying the example applications as part of your learning process, you should be aware that installing the v1.2 examples will overwrite the existing app files. At a minimum, you would need to reenter your database connection information. A few other nits include a situation (i.e. bug) where you now need to manually add a double quote to certain manually entered WHERE clauses, and a minor screw-up on SoftVelocity's part where they forgot to update the version comments in the templates themselves (they still say v1.10.) I only mention the last one in case you are compulsive (like

me) about verifying that an update was correctly installed. The out-of-date version comments (also displayed on the test pages) were confusing until SoftVelocity confirmed that it was just an oversight.

Late Breaking News: Clarion/ASP 1.3 Is Coming

Sigh

Just when I thought I was getting caught up, SoftVelocity told me they are currently prepping a new release. Here's an advance look at what v1.3 *could* include (note the use of the word "could" here – SoftVelocity reserves the right to alter this list before actual release):

- New Feature - Database connections can be specified at the procedure level. Note: The change in the database connection properties may require resetting the values for existing applications.
- New Feature - Global; Cursor location can be specified by procedure types (browse-form) / modes (insert-update-view).
- Enhancement - Generation of the Script engine version to the Test page.
- Enhancement - Added logic to consider the format options when populating JavaScript date functions.
- Change in behavior - Browse procedure; labels for custom columns are now static HTML.
- Change in behavior - The `LoadInclude` function calls are now at the start of the page merge routines to permit any tokens within the include to be replaced by the merge function.
- Change in behavior - Data connection properties; the runtime user and password prompts have been removed. The information is now taken from the connection string entirely.
- Fix - Generation of SQL `WHERE` statement missing ending quote under some circumstances.

Hmmm. Seems to be mainly new features, with the addition of a fix for the manual `WHERE` clause quote bug that popped up in v1.2. The first one, being able to specify the database connection on a per-procedure basis, sounds like a real winner. It should add even more flexibility for integrating multiple databases/data sources into a seamless user experience.

So when will it arrive? Version 1.3 should appear within a week or two of the publication of this article but, once again, SoftVelocity reserves the right to release things only when they feel they are completely tested and ready. In any case, let's hope upon the public release of v1.3 they also address some of the other nits I mentioned in my discussion of the v1.2 update process.

Embed points in Clarion/ASP

Ahhh, embeds. The lifeblood and the bane of Clarion Applications. Does Clarion/ASP have embeds? Sure it does – you just don't put Clarion code in them. What you do put in them depends on the type of embed it is, since a given embed, depending on where it is located, could have either ASP code (i.e., VBScript and ADO), HTML tags, JavaScript calls or even a mixture of all three (four, if you count ADO as a separate technology/syntax to learn.)

Yup, that's the scariest part of Clarion/ASP – you could potentially need to learn *four* more languages and/or technologies in order to take the system to its possible extremes. Backing off from that concept for a moment, here is a look at what some Clarion/ASP embed options look like in the IDE:

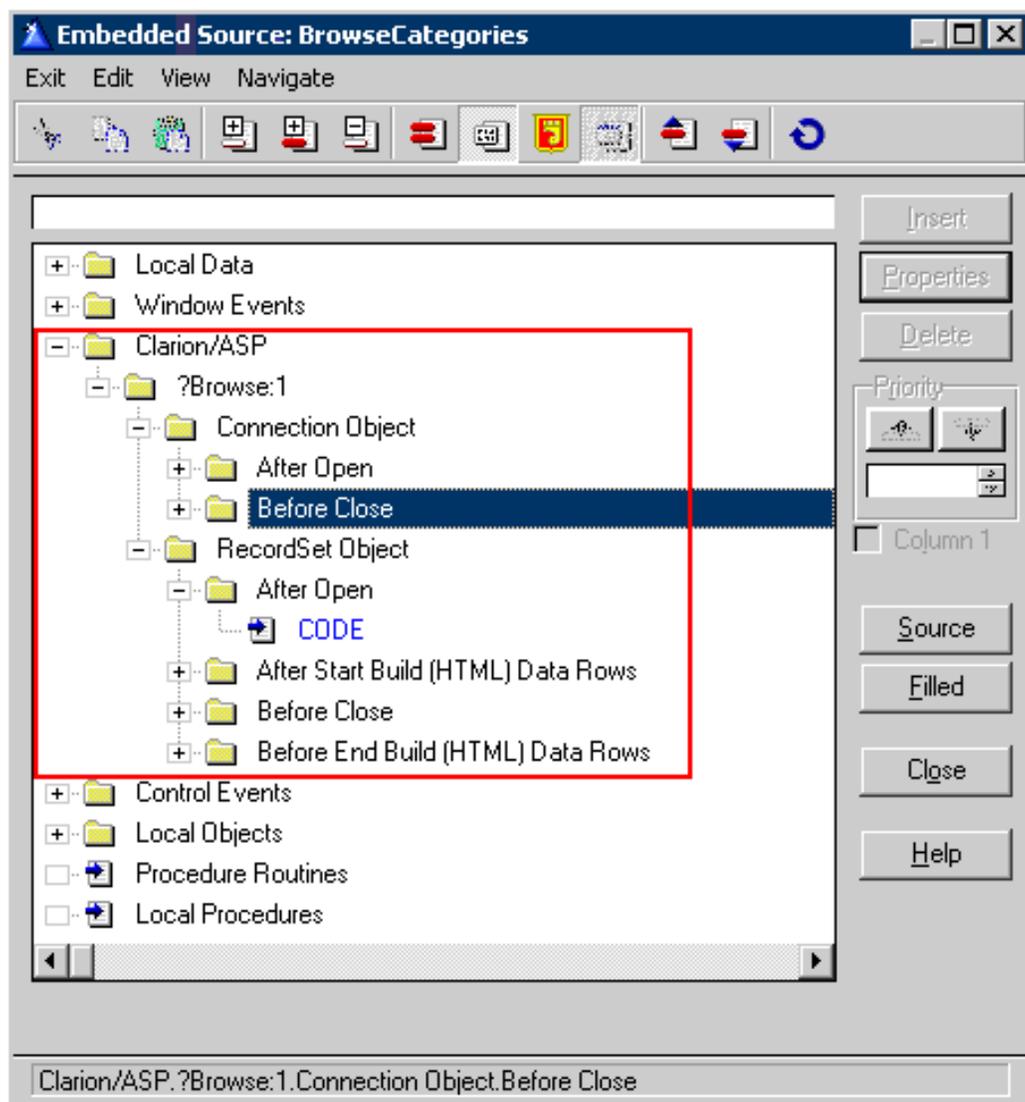


Figure 1. Embed points for a Clarion/ASP browse procedure.

Note that all of the standard embed points outside of the red square have no effect on the pages generated by Clarion/ASP. Conversely, anything you put in those Clarion/ASP embeds will have no effect on the generated Clarion source files.

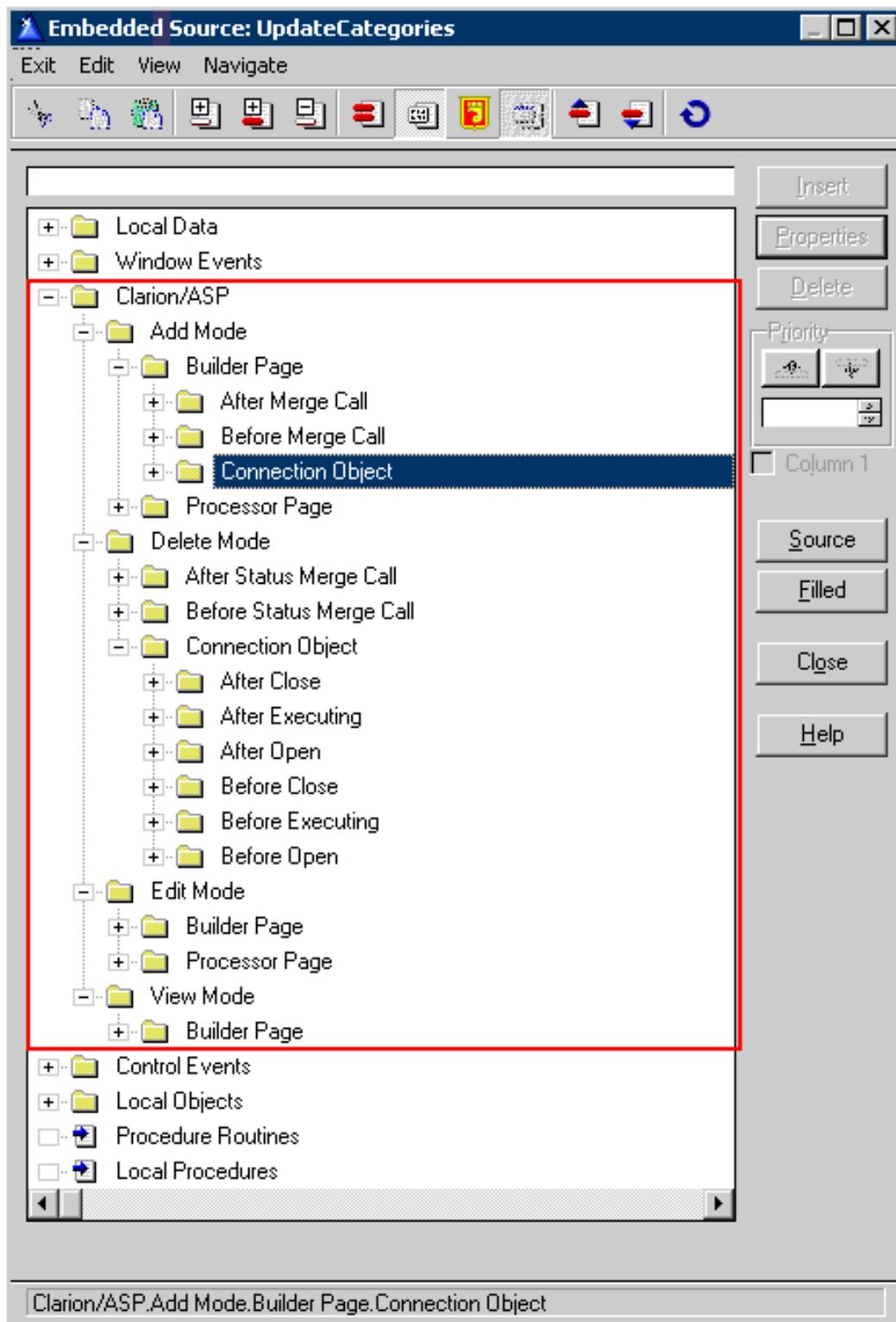


Figure 2. Embed points for a Clarion/ASP update procedure.

As you can see, the form procedure has more potential embed points than the browse. This is because Clarion/ASP will generate multiple ASP/HTML pages from a single update procedure – a pair for View, Add and Edit modes, plus an ASP Delete page.

Another thing to keep in mind is that the templates have all sorts of what I'd call pseudo-embed points – places within the templates themselves where you can embed bits of text (such

as HTML or VBScript functions) into the stream of generated code. These can range from the global include files used by `blank.htm` in all generated HTML pages to tiny snippets of HTML added to cells in a browse. Here is an example used for a custom link on each row in a browse:

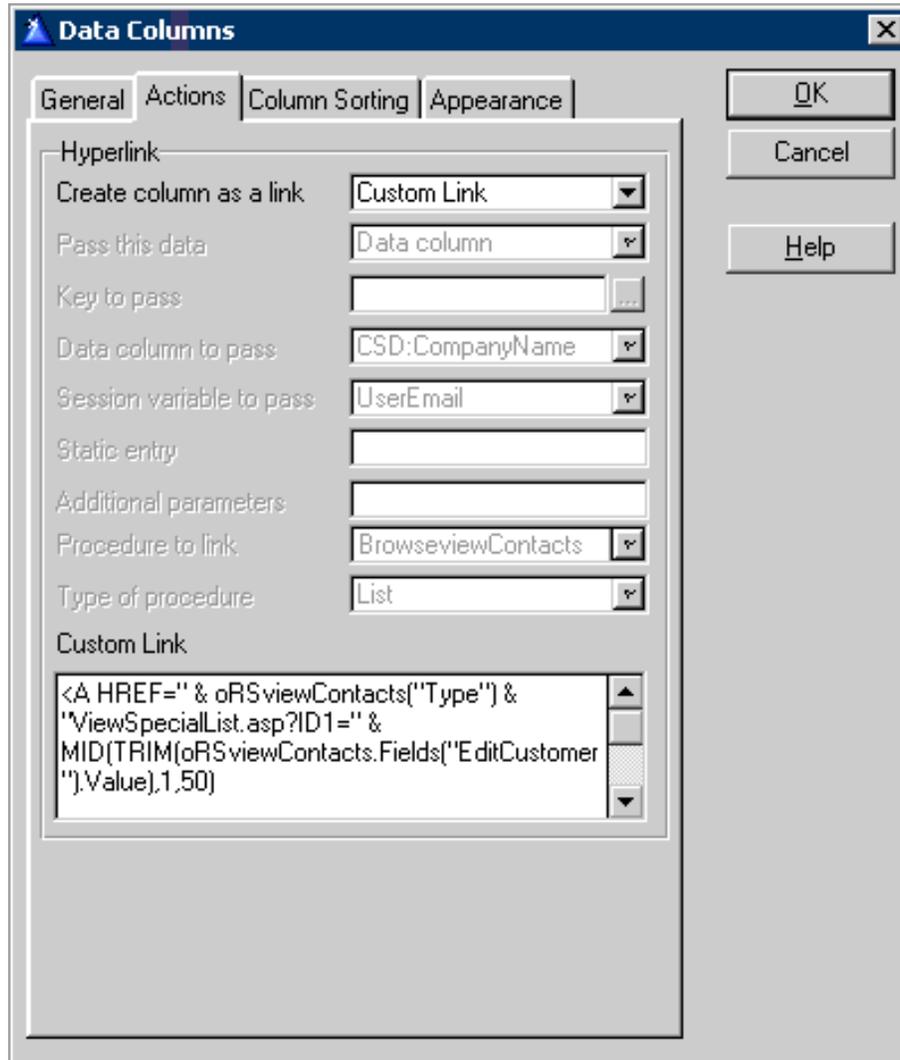


Figure 3. Embedding a combination of HTML and VBScript into a template prompt.

Since the code is truncated in Figure 3 by the word wrap in the template text box, the complete code snippet follows. Granted, this is a somewhat advanced example, but it will also serve to illustrate the types of things that can be done from within the templates.

```
<A HREF=" & oRSviewContacts("Type") & "ViewSpecialList.asp?ID1=" &
MID(TRIM(oRSviewContacts.Fields("EditCustomer")).Value),1,50)
&MID(TRIM(oRSviewContacts.Fields("EditSupplier")).Value),1,50)
&"><IMG alt='View Orders or Products' src='images/' &
MID(TRIM(oRSviewContacts.Fields("Type")).Value),1,50) &" 32x32.gif'
border=0></A>
```

In this case, the code will be used in generating the HTML for each row in a browse. Depending on the value in the Contacts record `Type` field, it will create a custom link

(including the appropriate image) to either a Customer edit page or a Supplier edit page. This particular snippet of code executes on the web server while the server is dynamically creating the browse page that will be returned to the user. By the time that page reaches the user's web browser, all you would see in the source for the page would be pure HTML links to the appropriate update page, i.e., the result of combining the output of the embedded VBScript and HTML.

OK, OK – let's get to the big question that I know is lurking in your mind right about now: Do you *need* to write embed code?

Just as in a Clarion application, the answer is no, *if* you are happy with the basic functionality provided by the templates. And just as in a Clarion Application, the likelihood of needing embed code increases as your requirements become more complex. The interesting point, though, is where you would normally cross that line from template-only development to needing embed code.

In a Clarion application, I'd say that line is pretty early in the game – I don't think I've ever shipped (or even built) a Clarion application that didn't have *some* kind of embed code somewhere. On the other hand, that line seems to come much later in Clarion/ASP for a couple of reasons.

First, because of the inherent limitations in the HTML/ASP/Web browser paradigm, there simply aren't as many ways to do things as there are in a straight Windows program. This means that it is more likely that the templates can anticipate your needs and let you accomplish your task from within them. As I said in a prior installment, these are very deep templates and there is a whole lot of functionality "in-the-box." The second reason is that the vast majority of changes that you will want to make will be made outside of the IDE.

Huh? *Outside* of the IDE? Here's why: The functional ASP code pages generated by the templates (the parts that connect to the database, and retrieve and navigate through record sets) look very clean to me, and they take care of the vast majority of the dirty work for you. This means there really aren't a whole lot of reasons that you would *need* to embed ASP/ADO code in the IDE. The place SoftVelocity can't anticipate your needs is in the visual design portion of your application – the HTML pages that the user views and interacts with. Sure, they give it a shot (just like in a Clarion application), but I doubt that many people would be totally happy with that type of generic output in either Windows or the web.

Now, since the Clarion IDE has no provision for visually editing HTML files, that means you'll be doing your work in both the design time and run time HTML files in an outside editor. Yeah, yeah – you *could* use the Clarion Editor to edit HTML source files, but then again, you could also write your Clarion programs without using any of the visual formatters in

the IDE, either. Not bloody likely, in my opinion. That means you'll need to do one of two things: get a visual editor (either commercial or shareware/freeware) and learn how to use it (along with learning basic HTML, etc.) or, find/hire someone else who already knows web design (and has the tools) and let them edit the HTML pages to achieve the desired look and feel.

Fortunately, the basic design of Clarion/ASP (splitting up the functional ASP code and the visual HTML pages) lends itself extremely well to this sort of division of labor. At the very least, I know from personal experience that it is much easier to get started editing HTML pages in a visual environment than it is to start writing ASP code by hand. The former lets you be productive with HTML without knowing the details, while the latter is strictly a know-what-you-are-doing-or-you-got-troubles affair.

In any case, when it comes to visual HTML editors SoftVelocity uses and recommends the market share leader, [Macromedia's DreamWeaver](#). Any other visual editor can be used as well – the primary thing you want to watch out for is editors that think they know better than you and reformat/hide tags to "protect" you from yourself (I'm talking about *you*, Microsoft FrontPage!). I'm also sure that SoftVelocity is aware that the lack of integrated visual editing is a drawback in Clarion/ASP and that they will address that need at some point in the future, hopefully by leveraging existing tools (the world does *not* need another HTML editor!)

To sum up on embed code, I think that most Clarion/ASP users won't use the formal embeds very often, if at all. A good visual editor to spruce up the HTML or create include files will pretty much be a requirement, though, if you are going to be the one doing site interface/user design and coding.

Integrating Clarion/ASP with existing sites

Recognizing that pages created using Clarion/ASP will not usually exist in a vacuum, SoftVelocity made it a design requirement that Clarion/ASP pages should be easy to integrate into existing sites, i.e., non-Clarion/ASP web pages should be able to easily call Clarion/ASP pages. To this end, there are three main areas where Clarion/ASP eases the task of integration.

Style Sheets

Clarion/ASP provides for relatively simple integration with the style sheets that might already be in use for a site. Global styles can also be overridden on a procedure-by-procedure basis.

HTML

There are two methods that help you to incorporate existing HTML into the output of the templates, and vice versa. The first is the HTML includes area of the templates. These let you

include existing site HTML files (usually just page fragments such as site wide page headers and/or footers rather than complete pages) into the standard blank.htm that all of the Clarion/ASP HTML pages are based on. From the opposite point of view, you can incorporate the %Clarion/ASP% symbol into a pre-existing page layout, use that page in place of blank.htm, and the Clarion/ASP merge process will merge the output of the ASP execution (a browse list, for example) into the site standard web page layout.

URL Parameters

Clarion/ASP pages provide for passing parameters as part of a URL. These allow you to provide required information to the page dynamically. Parameters that can be passed include ID1, used as, for example, the primary key value for selecting a record to update or as a filter for a browse list. Others include WHR, which lets you specify a filter explicitly, and HTMLT, which lets you dynamically specify which HTML page will be used as the based for the ASP merge (new for 1.2). There are others as well, but the point is that as long as external pages know how to call (and/or prime) the Clarion/ASP pages, they can all work together very well.

Also, if you have security turned on for the Clarion/ASP templates, any call to a secured Clarion/ASP page will automatically redirect the user to the login page. In other words, even though the non-Clarion/ASP pages on a site don't know anything about the internal Clarion/ASP security, your pages will still be protected.

Wrapping it all up

Well, it's been a long journey, beginning with the rather raw v1.0 and ending up at a much more versatile v1.2 (with another version coming soon.) During the time I have spent with Clarion/ASP, the product has been in a state of rapid improvement, with SoftVelocity paying a lot of attention to the problems and suggestions of the first users.

So what do I think of it? That's complicated, as Clarion/ASP is in no way, shape or form your normal Clarion add-on. Here are some of my conclusions:

- If you plan to go beyond the basics, expect that you are going to need to spend some time learning about all of the options in Clarion/ASP templates. As I said before, they are very deep with many levels of options and overrides for global options.
- Even though you can generate web pages with no knowledge other than how to use the Clarion IDE, you *will* need to spend some time learning about the basic technologies used by the pages it generates: ADO, HTML, VBScript and JavaScript. Of the four, the most important to know right off the bat are basic HTML and some of the common VBScript functions like TRIM() (similar to the Clarion CLIP function) and MID() (similar to SUB). Understanding these will let you take advantage of the many places in the templates where you can enter

- snippets of code to enhance the generated output.
- Do expect to do some HTML editing, and for that you will need to get a good visual editor.
- You'll also have to realize that you aren't writing programs for Windows anymore. It is helpful to understand the fundamentals of how server technologies such as IIS and ASP function, both in designing your web applications and setting your own systems up for testing (yes, more stuff to learn.)
- If you are thinking about buying Clarion/ASP, I would highly recommend getting it during the introductory period while SoftVelocity is including the online training. If you get it after that, I'd still say you should allocate for buying the training as an essential part of learning the product. The online classes and support groups are definitely worth it when you are starting out.
- Even though you can add Clarion/ASP templates to existing applications without impacting the Clarion/Windows side of things, I finally ended up deciding that it was cleaner to just create smaller apps dedicated to Clarion/ASP. It simplified the entire process, helping me to focus on just those sections of the application that were applicable to the web, and also let me bring the ASP templates more into the foreground by putting them on the main procedure properties window. Within a larger application, there is no real visual way to determine which procedures have Clarion/ASP attached and which do not.

So who is Clarion/ASP aimed at? If you were a new user looking for a web application generator, and didn't need Clarion/Windows applications, I'd have to say that it isn't the proper tool. 80% of the Clarion IDE has nothing to do with Clarion/ASP, plus the lack of an integrated visual HTML editor is a significant drawback. In that situation, I don't think it can compete (at is stands) with other dedicated web development systems. And if you are already an expert in another web development system, it would be a toss-up as to whether you would gain or lose from switching to Clarion/ASP – at that point you are talking personal preferences. It's not an all or nothing proposition, though – there is nothing to stop a person from using Clarion/ASP to take advantage of the strong data dictionary tools in the Clarion IDE and then integrating that data functionality into sites developed with, for example, MacroMedia UltraDev.

For the Clarion developer, however, it is a marvelous way to ease into pure ASP-based web development. You can begin slowly from within an environment you already are familiar with, as opposed to the shock of jumping into an entirely new application environment. For example, I have Macromedia UltraDev and a few others and, trust me, they do things in an *entirely* different way – not badly by any means, but differently. There is a lot to be said at times for staying in a comfort zone. (In case you haven't guessed, I have been tinkering with web development systems for some time, looking for the web equivalent of Clarion.)

Keep in mind, if you take the time to learn web design and development with Clarion/ASP, you are not really leveraging your Clarion/Windows knowledge – what you are doing is

learning a whole new business area where you can apply your hard-won knowledge of database and application design in new (and hopefully profitable) ways. And, as mentioned above, the separation of functionality between the database code and the visual presentation also lends itself perfectly to partnering with web designers who need database functionality but are, shall we say, data design challenged (not to mention the database gurus who are ‘visually challenged’ on the design front.)

Finally, most Clarion developers will probably originally look at Clarion/ASP as a way to make portions of their Windows applications available on the web (like I did). If you think about it, though, it can also be a powerful tool for the small developer to sell the other direction as well. Need a web application? Here it is. Oh, would you like a Windows version as well? Not a problem (rubbing hands gleefully). I think this is one of the areas where there are great potential synergies to be gained by using Clarion/ASP for web development. And I keep thinking about the upcoming Clarion/PHP templates as well. All of the time invested in learning the Clarion/ASP templates, web design and basic web technologies such as HTML should be directly translatable to Clarion/PHP, so I should already have a nice leg up on developing web applications for Linux. Sweet.

Bottom line: Clarion/ASP involves a whole new set of technologies for many Clarion developers, and there is simply no way to really take advantage of it without putting in some serious time to learn them. It isn't a silver bullet, but it *is* a heck of a system, an elegant design, and a very useful new tool for Clarion developers inclined towards the World Wide Web. I like it, I bought it, and I plan to use it quite a bit in the future.

Ability to do the task	
Ease of use	
Ease of Installation	 (as of v1.1)
Documentation	
Technical Support	
Black-Box DLLs/LIBs	No
Overall Product Rating as of v1.2:	

Clarion/ASP is available from SoftVelocity. I'd highly recommend going the [SoftVelocity web site](#) and browsing through their [Clarion/ASP](#) information, as there are numerous examples,

videos and sample sites built with Clarion/ASP to experiment with.

Pricing: As of the writing of this article the cost for the Clarion/ASP Templates is \$595. Clarion/ASP training is \$125 for the basic course, and \$200 for the advanced course. Online support with guaranteed responses from SoftVelocity and access to dedicated newsgroups is priced at \$150 for six-months. Bundled pricing may or may not still be available – contact SoftVelocity sales for current information.

According to SoftVelocity, patches will be made available to all owners of Clarion/ASP even if they are not currently signed up for the dedicated support package.

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

Reader Comments

[Add a comment](#)

Copyright © 1999-2002 by [CoveComm Inc.](#) All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited.

INVEST
in your own abilities**Clarion**
magazine[Ads](#) · [Comments](#) · [Writers!](#) · [Privacy](#) · [Contacts](#) · [PDFs](#) · [Freebies](#) · [Open Source](#)[Topics](#) > [News](#) > [ClarionMag 2001 News](#)

Clarion News

Published 2001-11-21

[Office Templates Gold Release](#)

Office Templates v2.0 is now available, and are still freeware. A new commercial set of classes based on the free classes is now in development.

Posted Monday, October 28, 2002

[50% Off Gitano's gCal](#)

Gitano's gCal is now 50% off. Version 3.0 includes nine calendars, a date calculator, a control calendar (add to your procedure, not a pop up), time picker, 35 date/time functions, quick keys, plus much more. Implementation is as simple as adding a global template to your application.

Posted Monday, October 28, 2002

[EntabberPro Enter=Tab Solution](#)

EntabberPro let's you fine-tune the behavior of the enter key, including: Define on which control types you want enter replaced and on which not; Choose the applications for which to enable enter=tab conversion; Optionally only acts on the numeric keypad's enter key; Ctrl+Enter generates a regular enter key stroke; Global hot key to enable/disable EntabberPro. Entabber Pro does not require any changes to your application, and will also work for any other programs. Free 30 day trial. Licensing options include end user licenses, or redistributable developer licenses. EntabberPro is available at ClarionShop (<http://www.clarionshop.com>).

Posted Monday, October 28, 2002

[XScan 1.2 Released](#)

Version 1.2 of XScan is now available. This version has improved stability of data migration interfaces and better support for copying from ISAM files to SQL tables. XScan is a free, light yet powerful tool to view, edit and manipulate any data files recognizable by Clarion. XScan is based on XLib library and is able to handle files/tables with a structure defined at runtime.

Features include: Editing; Mass updates/inserts/deletes; Data migration/export/import from any format to any format; Ability to parse Clarion source to obtain declarations; JOINed VIEWs, the files/tables can belong to different databases or use drivers; Multi-synchronization of several files/tables.

Posted Monday, October 28, 2002

[SealSoft xFText 1.0](#)

SealSoft has released xFText v1.0, a class and extension template that lets you add text anywhere on your application's frame. Source included. xFText supports single EXE, multi DLL (Local/Standalone), 32 bits. You can set all text attributes include normal, light, and dark colors for 3D effects. Demo available. Price is \$39.

Posted Monday, October 28, 2002

[Browse Header Sort And Reverse Sort Template](#)

Carlos Gutierrez has announced a browse header sorting template for ABC. This is an extension that you add to the browse. Comments welcome.

Posted Monday, October 28, 2002

[xSmartMacro v2.2 Now Freeware](#)

SealSoft's xSmartMacro v2.2, which enables macros via the clipboard, is now freeware. Features include: Tree-like storage of macros; pasting macros into Clarion embeds with a single click; System tray popup menu; Export/import macros; Search feature; #PROMPT-like capabilities; Flexible end user setup.

Posted Monday, October 28, 2002

[ExcelBond 2.0 Released](#)

ExcelBond 2.0 supports importing from and exporting to Excel files. This is an add-on for the IMPEX templates. Demo available.

Posted Monday, October 28, 2002

[Gitano Software Affiliate Program](#)

Gitano's new affiliate program lets you earn money from any sale of a Gitano product generated from your web site. Have as much or as little information of your web site as you want. Gitano will provide images, banners, and links as needed. If you are interested please send a private e-mail to jfmoreno@gitanosoftware.com.

Posted Monday, October 28, 2002

[Clarion Third Party Profile Exchange Update](#)

There are 396 Product Profiles and 333 Vendor Profiles in this release. You must have Product Scope 32 PRO Version 4.5 to view profiles with data files.

Posted Thursday, October 17, 2002

[German Clarion Distributor Clearing Stock](#)

Downsizing Systems, the German Clarion distributor, is clearing out old stock. This includes a bunch of tools for Clarion/DOS as well as for Clarion/Windows. Many of these products are 'for internal use' only so not updatable or for commercial use. The products are in English, except CA Clipper and a German (!) Clarion for DOS. These are for 95 % single pieces, so act fast. Click in the column 'Neuigkeiten' on the link 'Aktion: Platz im Lager' (in the left upper area of the page) and you will see a list with the product, vendor, description (though in German). The prices are in Euro, 1 Euro is about 1 US\$, so easy to calculate. The prices are from 5 US\$ up. None of the products for sale are 'Not for resale' products or current versions for low money. The majority of the products are old versions.

Posted Thursday, October 17, 2002

[EasyExcel And EasyReport Reviews At ClarionShop](#)

There are two new reviews available on the ClarionShop review page: EasyExcel has been reviewed by Mitchell Alexander, and EasyReport has been reviewed by Johan de Klerk

Posted Thursday, October 17, 2002

[INN Bio for 15-Oct-2002](#)

This week, the Ictips News Network is pleased to present a bio that many have requested. One of the most well-known and technically able programmers among us, he's actually a Harvard-educated dentist and retired Air Force officer. He has a dry sense of humor, and it comes through in this interesting bio (even if listening to him talk about programming gives me an inferiority complex ;). And he has some great stories, too.

Posted Thursday, October 17, 2002

[New Version Of BoTpl](#)

A new version of BoTpl is available for downloading. New features include: Updated Clarion.syn file for textpad with lots more symbols; BoAppInfo has new features and one bug fix; Greg Berthume contributed Window Labels, category limiting of procedures, just reports, etc., and Long Procedure description; Three quick option set buttons for all the features

Posted Thursday, October 17, 2002

[XP Menubar Version 3 Released](#)

The latest version of XP Menubar has been emailed to everyone on the list. Refer all comments and questions to the Open Clarion Newsgroup at news://news.openclarion.org/ and use the newsgroup openclarion.community.general.

Posted Thursday, October 17, 2002

[EasyMultiTag 2.03](#)

New in EasyMultiTag ver 2.03: Turn Windows style tagging mode on/off at runtime; Turn

exact tagging mode on/off; Tag All; New control template for EMT; New demo; Bug fixes. This is a free upgrade for all registered users.

Posted Thursday, October 17, 2002

[CPCS Support delay](#)

CPCS support will be unavailable from Monday, Oct 14, 2002 through Friday, Oct 18, 2002. All requests for support (both in the newsgroups, and via email) will be handled as quickly as possible after this period.

Posted Monday, October 14, 2002

[Replicate Sneak Preview](#)

CapeSoft is in the final stages of completing an alpha release of its new Replicate product. Replicate provides an automatic, driver independent, file-version independent mechanism for replicating the data in two or more databases. Replicate logs your changes, adds and deletes and then using a transport manager of your choice, exports the changes to the other sites, where the changes, adds and deletes are imported into that data set. This all happens on the fly, behind the scenes. Replicate supports both offline and online environments appropriately.

Posted Monday, October 14, 2002

[New Updates From Gitano Software](#)

The following products have been updated and are now available free to registered users: Look Good Package; Theme Pack; gBuddy. Two new downloads are available that combine the themes from the Theme pack and gBuddy. These downloads do not contain the image files, just the COLOR.TPS file and a sample application. Note: You must have gBuddy and either the Theme Pack or The Look Good Package.

Posted Monday, October 14, 2002

[gSec At 50% Off](#)

For a limited time you can get gSec at half price. This offer is good for the full version, upgrades and competitive upgrades when you purchase directly from our secure online store: Full Version reg. price \$149, now \$74.50; Competitive upg. price \$99, now \$49.50; gReg Users upg. price \$80, now \$44.50.

Posted Monday, October 14, 2002

[Free TPS Replication](#)

Jerry Ray's TPS replication code is now available at no charge.

Posted Monday, October 14, 2002

[ImageEx Review At ClarionShop](#)

A review of ImageEx version 1.3, by E. Keith Neely, is now online at ClarionShop.

Posted Monday, October 14, 2002

[Credit Card Number Verification Algorithms](#)

Carlos Gutierrez points out this page as a source of credit card number verification algorithms. Keep in mind that these only tell you that the card number is valid, not that the card itself is valid!

Posted Monday, October 14, 2002

[SimPad Templates](#)

Eric Churton has released the SimPad Templates, which allow you to create your own keypads for touchscreen (or normal) applications. These are ABC only, C5 and C55. An easy-to-use program allows you to design your screen, optionally drag-n-drop images onto the buttons, set text and font, and also automatically set which keypress will be sent when the button is set. Then a utility template lets you import the Window you have designed, with all its functionality, into any existing application you have. The code generated shows in the white (editable) portions of the sourcecode as if you had typed it in yourself, and can be changed, deleted or enhanced as you see fit. The price of the SimPad templates is \$49 US. The existing Simsoft Templates are priced at \$69. A bundle of both the Simsoft and SimPad templates is available for \$99. For existing users of the Simsoft Templates however, the upgrade to the bundle is US\$30.

Posted Monday, October 07, 2002

[ThinkData Announces OutlookFUSE COM Automation For Microsoft Outlook](#)

ThinkData, Inc. has announced the official release of OutlookFUSE, a native COM (Component Object Model) automation interface to Microsoft Outlook using Clarion 5.5. This product supports both native early and late binding. OutlookFUSE relies on the professional version of the Plugware COM classes. It has been designed to support Outlook 98, Outlook 2000 and Outlook 2002 (XP). Example app is available. Cost is US\$199. Other products currently in development include: cdoFUSE, QuickBooksFUSE, and WinFaxFUSE.

Posted Monday, October 07, 2002

[Firebird 1.5 Alpha](#)

Kelvin Chua reports that the Firebird 1.5 Alpha is now available.

Posted Monday, October 07, 2002

[Simsoft Templates Update](#)

The Simsoft Templates now include a template to allow for global setting of field and button highlighting on selection. The functionality of the existing highlighting template has been adjusted so that when you set a global color you can still override the color in individual procedures (if you ever need to). Also included are several more Keyboard options, (which had been requested by an existing Simsoft user) that allow one to set the size (font) for keyboards

for individual procedures rather than once globally, and to set the xy starting position and optionally make them unable to be moved by the user. These upgrades are free to registered users. You should already have been notified of this, but if not, the updates can be obtained from www.clarionshop.com.

Posted Monday, October 07, 2002

[IceTips Newsletter](#)

The first IceTips Newsletter is now together and available on the IceTips website. It has some information about what's been happening at IceTips, and what will happen in the future. There is also some technical information, including a very interesting use of the CHAIN command, and a trick for adding records to a database with a single mouse click.

Posted Monday, October 07, 2002

[New XPMenu Demo](#)

A new XPMenu demo is now available

Posted Monday, October 07, 2002

[INN Bio & News for 1-Oct-2002](#)

This week's INN Bio features a different kind of Clarionite. She may not write programs with Clarion, but her work is vital to Clarion's biggest third-party vendor, and as such, she's known to many people in the Clarion community. A former legal secretary, now an "impersonal assistant", she keeps ClarionShop on track.

Posted Monday, October 07, 2002

[IceTips Bio Notifications](#)

Get on the IceTips bio notification list. IceTips bios, marketing moments, and news items that Sue writes will be published on a new bi-weekly schedule: watch for updates every other Tuesday.

Posted Monday, October 07, 2002

[Freeware SMTP MDA](#)

Ville Vahtera has released `vv_smtpagent`, a mailer library which sends messages and file attachment directly from your PC to the SMTP mail server (no need for MAPI). Your application will need access to the port 25. If you're operating within a restrictive firewall then you won't be able to use this library. This happens mainly with the larger free ISPs and intranets.

Posted Monday, October 07, 2002

[PHP/SQL Info](#)

Carl Barnes recommends SitePoint for some good articles on PHP and SQL. If you're looking forward to Clarion/PHP, you may want to do some reading now. There is lots of other web

info at this site as well, including ASP, XML, HTML, etc.

Posted Monday, October 07, 2002

[Free Wallpaper](#)

Gitano Software has some free wallpapers for applications and the Clarion IDE. Scroll to the bottom of the page.

Posted Monday, October 07, 2002

[CapeSoft Incremental Upgrades](#)

A number of CapeSoft products have had recent incremental upgrades, including NetTalk (2.3e beta), Insight Graphing (beta 7i); Draw (beta 10); MessageBox (1.6f); and HyperActive (1.7c).

Posted Monday, October 07, 2002

[Upcoming CapeSoft Products](#)

Coming soon from CapeSoft, Office Inside. This project wraps up the various Microsoft Office components and makes them available to your application. First is MS Word, then Excel, Outlook and so on. Built on the ultra-fast, and ultra-stable Plugware COM classes (Clarion 5.5 only). Also in the works is Replicate, which allows your Clarion program to replicate data between sites. It is a file driver, and file version independent.

Posted Monday, October 07, 2002

[File Manager 3 In Beta Testing](#)

After a short alpha program, File Manager 3 is now firmly in beta testing. The MsSQL driver, and Oracle driver support is currently active. Lesley is currently doing the ODBC driver, with MySQL and Interbase the main targets there. FM3 is being marketed as a separate product to FM2, however as it includes the existing FM2 engine CapeSoft recommends new purchasers to go straight to FM3. FM2 will still be supported for those that don't want to cross-grade. For a limited period only, FM3 is on special, down from the regular price of \$199, to \$149. FM2 users can cross-grade for \$99.

Posted Monday, October 07, 2002

[Comm Ports, Excel Code, Etc.](#)

Igor Kuklin has a number of Clarion code examples available for download, including batch emailing, reading/writing Excel files, and reading/writing COM ports.

Posted Monday, October 07, 2002

[Powerbrowse Site](#)

A reminder, courtesy of Edward Loh, that updated versions of Powerbrowse are available from Alan Telford.

Posted Monday, October 07, 2002

Reader Comments

[Add a comment](#)

Copyright © 1999-2002 by [CoveComm Inc.](#) All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited.

[Topics](#) > [Non-tech](#) > [ClarionMag Commentary](#)

Not Having Six Makes Me Cranky

by **Carl Barnes**

Published 2002-10-15

Editor's Note: The Cranky Programmer has been suffering from a bout of contentment lately, but appears to be on the road to recovery. During Cranky's convalescence, Clarion Magazine is pleased to present the commentary of other disaffected developers.

Romeo lamented, "What's in a name, would a rose by any other name not smell as sweet?" Romeo suggests that the beauty (or value) of a thing is not affected by what you call it. If a rose was named "IdontknowwhatIdidbutI'mreallysorry" would it not still get me out of trouble? (Some think that Shakespeare was poking fun at his competition the Rose Theater and its less-than-pleasant-smelling sanitary arrangements.) But enough about smells and Shakespeare, this is a Clarion journal. I say calling the next Clarion version 5.6 does affect its perceived value, in a negative way. And that makes me cranky!

Why must SoftVelocity call the next release 5.6? Some say it's because at DevCon98 Bruce Barrington promised that version 6 would be 32-bit. As good luck would have it, I was there and I still have my notes. Bruce did *not* get up like Julius Caesar and make an announcement "Friends, Romans, Countrymen, lend me your ears for 6.0 will be of 32 bits." Bruce said it during a Q&A session at the end of the show. It was always made clear at DevCon that many statements made were tentative plans, inside info, subject to change, etc. By the way, Bruce's Q&A session was named "Clarion Confidential." Get the point?

I remember something else Bruce said. At the DevCon where TopSpeed announced Clarion 4, he discussed and regretted the way they had numbered some major versions: CPD 2.1, CFD 3.1, CW 1.5. He said "Geez! The way we've been assigning these numbers you'd think we had to pay for them!"

SoftVelocity sees the "6 will be 32-bit" statement as a mandate. That makes me cranky. While the exact words were "6.0 will be 32-bit," what was under discussion and what was really meant was the *next* version of Clarion will be 32-bit. If you look at it that way, then SoftVelocity is not meeting the mandate.

But so what? That does not make cranky. I spend little time wishing the IDE were 32-bit. The IDE works very well for me every day. And I, like most of you, earn my living by what I produce in the IDE. That Blue Cloud has a silver lining! If SoftVelocity can't do a 32-bit IDE this time around that's fine with me. I'd rather have it done later, done well and see it called version 7 or 8.

In the next version of Clarion, SoftVelocity will release many enhancements and features that were not even under consideration at DevCon98: a new threading model; some great new template stuff; stuff that will save me time; stuff that is needed more than a 32-bit IDE. Good Stuff! I don't sell my customers the IDE, I sell 'em *stuff*.

The biggest problem I have with calling it 5.6 is selling that to my customers. They all know their software is developed in Clarion. When a new Clarion version comes out I have to sell my customers on paying me to upgrade their software. A point one update from 5.5 to 5.6 sounds like a minor patch that should not cost much, or be free, but the new threading model will require a careful review of code and complete testing. It will be an easier sale for me if I can say "I'm upgrading you to the new version 6." Then I do not have to explain to them "the 5.6 story."

It also looks bad to my customers because SoftVelocity does not *appear* to be updating Clarion in any major way. Other companies release new versions each year, why not Clarion? Versions 5.5 and 5.6 get mentally rounded to version 5 and we've been on 5 since 1998. This decimal version naming makes Clarion look like a dying tool which needs to be replaced by something that is kept updated. Since Clarion is my tool that means I get replaced!

The second problem I have with a "5.6" release is I just hate the sound of it. It gives me a melancholy pain. To me Clarion is a dream developer tool. The coming version has some stuff I've been dreaming about, and 5.6 just doesn't sound like "such stuff as dreams are made on." If I have to pay for this update, I will pay gladly, but I want more than a point one increase!

SoftVelocity, please forget what was said four years ago and stand up and take a bow for a great new version. Take the number you deserve, 6. Or call it 7 because 5.5 should have been 6. From what I saw at ETC everyone will be so happy with all the new stuff they'll forget about 32-bit. Or as Shakespeare would say, "All's well that ends well."

And if you call it 5.6 then I say "Unnatural deeds do breed unnatural troubles."

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

Reader Comments

[Add a comment](#)

Hey, Dave! How can I change my vote? Carl has convinced...

Have a look...

When preparing the article I did find my DevCon 97 notes...

I agree wholeheartedly, the upcoming release should be 6.0...

I think using a NAME might be an option - this is where...

One consideration is file naming of the DLLs. I currently...

It's a mistake for SoftVelocity to focus so much on the...

I agree. If it helps here is my vote for 6.0 (even if it...

Why not call it 15.0? Just having a number does not...

I throw in a vote for 6.0 as well! Why? Well I really...

Copyright © 1999-2002 by [CoveComm Inc.](#) All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited.

For marketing your Applications
and Developer Accessories or to
purchase other 3rd Party Tools ...

Developer **PLUS**™

[Ads](#) · [Comments](#) · [Writers!](#) · [Privacy](#) · [Contacts](#) · [PDFs](#) · [Freebies](#) · [Open Source](#)

[Topics](#) > [Tips/Techniques](#) > [Clarion Challenge](#)

Clarion Challenge Results: ContainsMatch

Published 2002-10-17

My thanks to all who entered. The challenge, while simple enough (thankfully, as any complex challenge would be horrendous to judge), threw up the usual confusions! In particular, many entrants were uncertain about the characters to ignore (the lesson learned was to write a more complete spec, and keep meddling editors away!). To this end any entry which failed the initial tests (and arrived before the deadline) was permitted one resubmission (my apologies to Ladrière Xavier whose email bounced and thus didn't get the second chance).

Okay, on to the challenge. At its core the problem can be summed up as "Does 'xyz' contain ALL the chars 'abc'." The actual challenge was more specific (to ignore all chars except "a-z, A-Z, 0-9"), but the heart of the problem remains the same.

Analyzing Algorithms

One of the more popular measures of algorithms is the "big-oh" notation ("O").

Definition: A theoretical measure of the execution of an algorithm, usually the time or memory needed, given the problem size n , which is usually the number of items.

For example: For some function f , taking n as its input, the equation $f(n) = O(g(n))$ would mean that it take less than some constant g times n . This is expressed as " f of n is big oh of g of n ". The following procedure adheres to this example:

```
getLeapCount procedure(ulong theDate)
  code
  loop i = 1 to theDate
    !do the magic
  end
  return ...
```

Here "do the magic" will be some const value g and will execute n times. One nice thing about the "O" notation is that it says to just ignore the trivial stuff and focus on the main "decelerator" of the algorithm (a loop for example).

The most common examples are (in order of speed)

- $O(1)$ - Not dependant to any variable number, i.e. a constant time
- $O(\log N)$ - A binary chop is an example of this. Remember the old game – pick a number between 1-100, then as the one person guesses the other says higher or lower, so getting to 84 would look like 50-higher, 75-higher, 87-lower, 81-higher, 84 - you got it!
- $O(N)$ - Proportional to the size of N (this would include a loop to N , also loops to any constant multiples of N ($0.5N$, $2N$, $100N$)). So if you double N the function would take twice as long.
- $O(N \log N)$ - Similar to the binary chop, but also doing N activities for each iteration (a merge sort for example).
- $O(N^2)$ - Proportional to $N*N$. So if you double N , the function would take four times longer. Two nested loops dependent on N would yield this result.
- $O(N^3)$, $O(N^4)$, ..., $O(N^n)$ - (triple/quad nested loops etc.).
- $O(N!)$ - which is a function which slows down exponentially on the size of N .

The above of course are generalities. It is easy to code a $O(N)$ function which is slower than a $O(N^2)$ function. In fact the challenge showed this nicely – anyone who used a QUEUE came last (even when their algorithms were more "efficient") – the analogy being why would you use your car to get from your bed to the toilet?

NOTE: For an expanded discussion of the various equations see Mark Dunlop's ["A Rough Guide To Big-Oh Notation."](#)

To the test

Right, time to look at the entries. Since the CompareMatch function takes two main parameters, `txt` and `chars`, I will be using N and M to represent them. Basically all the entries fell into two categories, the $O(N+M)$ and the $O(NM)$. Since LOWER, UPPER, INSTRING, memchr, and CLIP are all $O(N)$ functions, the following typical code falls into the $O(NM)$ category (bad):

```
loop i = 1 to len(clip(chars))
  if not instring(choose(noCase = true, lower(chars[i]), |
    chars[i]), choose(noCase = true, lower(txt), txt), 1, 1)
    return false
  end
end
return true
```

To prove the point I took the best $O(N+M)$ solution and the best $O(NM)$ solution and put them head to head on some small numbers:

N	M	Winner
1	1	$O(N+M)$
2	1	$O(N+M)$
1	2	$O(N+M)$
2	2	$O(N+M)$

The test was a bit disappointing as I was hoping the $O(NM)$ solution would be quicker for the very small cases; the results where, however, very, very close!

It's also worth noting that the results where very different in debug mode versus release mode; naturally the final results are based on release mode.

The following results only include functions that passed the test (see the source code for a complete list). The scores are the percentage of the speed of the overall winner:

#15 – John Griffiths (score: 2)

Nothing wrong with Johns approach, it was basically an $O(N+M)$ solution, but he used a QUEUE to store his character hits, had he used an array, he would have been easily in the top half.

#14 – Matt Grossmith (score: 10)

Matt's approach was very nearly a $O(N+M)$ but ended up a $O(NL + ML)$, where L was a list of valid chars (note the double loop because of the inner INSTRING):

```
...
StringLen# = len(clip(P:Chars))
loop char# = 1 to StringLen#
  SingleChar = choose(P:Nocase=True, |
    upper(P:Chars[ char# : char#]), |
    P:Chars[ char# : char#])
  CharPos# = instring(SingleChar, MasterMap, 1, 1)
  if CharPos#
    SrchMap[CharPos# : CharPos#] = SingleChar
  .
...

```

#13 – Ville Vahtera (score: 12)

Ville used a classic $O(NM)$ solution:

```
...
Loop x# = 1 To Len(Tempchars)
  Loop y# = 1 To Len(Temptxt)
    If Val(Tempchars[x#]) = Val(Temptxt[y#])
      RetVal = RetVal + 1
      break
    End
  End !second loop
End !first loop
...
```

#12 – Stephen Bottomley (score: 12)

Stephen (not unlike Matt) had a $O(N+M)$ solution, but used an inefficient test for invalid chars and ended up a $O(L + N + M)$, where L was in fact three loops nested inside a 255 loop (INRANGE is a loop after all), thus pushing his solution down to this position:

```
...
loop Counter = 1 to 255
  if not (inrange(counter,48,57) or |
          inrange(counter,65,90) or |
          inrange(counter,97,122))
    CharArray[Counter] = 2
  .
.
...
```

#11 – Brice Schagane (score: 17)

Brice's solution was, to me, the obvious solution, and the one I thought I would have seen more of. While small and simple, Brice's code ended up being in the $O(ML + MN + MM + MN)$ category (due to two inner INSTRINGS, and two inner UPPERS). The UPPERS could have been easily moved outside the loop:

```
...
LOOP myPos = 1 TO LEN(xChars)
  myChar = xChars[myPos]
  IF NOT INSTRING(myChar, myQualifiers, 1, 1) Then CYCLE.
  IF xNoCase Then
    myChar = UPPER(myChar)
    xText = UPPER(xText)
  END!_IF
  IF NOT INSTRING(myChar, xText, 1, 1) Then RETURN FALSE.
END!_LOOP
RETURN TRUE
...
```

#10 – Jeff Berlinghoff (score: 40)

Jeff's approach was the same as Brice's, without the two inner UPPERS (look at the difference that made) – this made it a $O(ML + MN)$ solution.

#9 – Lew Strook (score: 44)

Lew's code was the same as Jeff's, but without the fancy and unnecessary NEWing of strings.

#8 – Geoff Robinson (score: 48)

The code is getting fancy now! This is the first entry using the clib function memchr, which is basically a variant of INSTRING but only finds a single char in an array of chars. Geoff's code is a $O(NM)$ solution, but probably slower than the other memchr solutions due to the use of the CASE string code. CASE is notoriously slow with strings (although fine with numbers):

```
...
LOOP i = 1 TO L:LenChars
  TestChar = chars[i]
  CASE TestChar
  OF 'a' TO 'z' OROF 'A' TO 'Z' OROF '0' TO '9'
    if ~MemChr(address(txt),Char2,L:LenTxt) |
      then return FALSE. ! exit if no match
  END !case
END !loop
...
```

#7 – Brian Ekeland (score: 68)

Brian supplied a $O(NM)$ solution, but the second fastest at that, basically because it is using a sensible conditional to filter out the unwanted chars:

```
...
loc:byte = loc:charsbyte[loc:charsndx]
if loc:byte < 48 |
  or (loc:byte > 57 and loc:byte < 65) |
  or (loc:byte > 90 and loc:byte < 97) |
  or loc:byte > 122
  cycle
end
...
```

#6 – Carl Barnes (score: 75)

Carl had two entries, a $O(NM)$ and a $O(N+M)$. I included both (this is the $O(NM)$) as Carl had stated a preference for the $O(NM)$ because it was faster in his tests, and indeed it is the fastest $O(NM)$ solution in this group. I disagree with Carl as his $O(N+M)$ was faster in my test cases.

#5 – Phil Will (score: 86)

Phil had two entries, nearly identical. I included them both as they produced a 5% difference (and two places in the test). This one is a nice $O(N+M)$ solution, local vars do *not* have the AUTO attribute and there is one extra use of VAL. I suspect, and maybe Phil will (no pun intended) validate, that it is the lack of AUTO that made the difference.

#4 – Mark Goldberg: (score: 87)

Mark's approach at first glance looked expensive to me due to the following code:

```
...
loop CurrByte = 0 to Val('0') - 1
  TxtContains[ CurrByte ] = 1
end
loop CurrByte = Val('9') + 1 to Val('A') - 1
  TxtContains[ CurrByte ] = 1
end
loop CurrByte = Val('Z') + 1 to Val('a') - 1
  TxtContains[ CurrByte ] = 1
end
loop CurrByte = Val('z') + 1 to maximum(TxtContains,1)
  TxtContains[ CurrByte ] = 1
end
..
```

But if you inspect the above closely you will find it is a < 192 iteration loop and an efficient one at that, making his procedure a $O(L + N+M)$ procedure, where L was small.

#3 – Carl Barnes (score: 88)

Carl is back with his $O(N+M)$ solution!

#2 – Phil Will (score: 92)

Phil is back with his "clean" $O(M+N)$ version. This version also gets my stamp of approval on readability, as it very clearly does what it does (apart from the use of "1" for true and uppcased keywords – but hey, the latter is subjective):

```
...
IF noCase
  txt=UPPER(txt)
  chars=UPPER(chars)
END
CLEAR(TxtArray)
! Load Text Array (ignores 0)
LOOP I=1 TO LEN(CLIP(Txt))
```

```

    TxtArray[Val(Txt[I])]=TRUE
END
! Check VAL
Found=1
LOOP I=1 TO LEN(CLIP(Chars))
    ThisChar=VAL(Chars[I])
    CASE ThisChar
    OF   eValZero TO eValNine |
    OROF eValA    TO eValZ    |
    OROF eVala_   TO eValz_   |
        IF TxtArray[ThisChar] THEN CYCLE.
        Found=FALSE
        BREAK
    END
END
RETURN FOUND
...

```

Drum roll please...

And the winner is...Adriaan van Ieperen (score: 100)

Adriaan's entry (a $O(M+N)$ solution obviously) is over 8 % faster than his nearest rival:

```

ReturnValue          BYTE(TRUE)
cArray              BYTE,DIM(256)
c                   LONG,AUTO
CODE
! loop 1: mark usage of all characters in txt string in cArray
LOOP c = 1 TO LEN(txt)
    IF noCase THEN
        ! check if char is between 'a' and 'z'
        ! convert to uppercase and mark usage
        IF VAL(txt[c]) >= 97 AND VAL(txt[c]) <= 122 THEN
            cArray[VAL(txt[c]) - 32] = TRUE
        ELSE
            cArray[VAL(txt[c])] = TRUE
        END
    ELSE
        cArray[VAL(txt[c])] = TRUE
    END
END
! loop 2: check usage of chars that
! need to exist in txt string
! if a char is not used then set returnvalue
! to false and break out of loop
LOOP c = 1 TO LEN(chars)
    ! filter out all chars that are not
    ! between 'A' to 'Z', 'a' to 'z'
    ! and '0' to '9'
    IF (VAL(chars[c]) >= 97 AND VAL(chars[c]) <= 122) OR |
        (VAL(chars[c]) >= 65 AND VAL(chars[c]) <= 90) OR |
        (VAL(chars[c]) >= 48 AND VAL(chars[c]) <= 57) THEN
        IF noCase THEN

```

```

    IF VAL(chars[c]) >= 97 AND VAL(chars[c]) <= 122 THEN
        IF NOT cArray[VAL(chars[c]) - 32] THEN
            ReturnValue = FALSE
            BREAK
        END
    ELSIF NOT cArray[VAL(chars[c])] THEN
        ReturnValue = FALSE
        BREAK
    END
    ELSIF NOT cArray[VAL(chars[c])] THEN
        ReturnValue = FALSE
        BREAK
    END
END
END
END
RETURN ReturnValue

```

Why is Adriaan's code so fast? There are two main reasons:

1. **Very few calls into the c55runx.dll.** LEN and VAL are the only functions called. When code is this tight, calling other procedures becomes an issue.
2. **Simple IF statements:** Adriaan tells the compiler exactly what to do. It might produce more machine code for each path, but each path does exactly one thing.

Summary

I hope these entries have provided some new ideas on how to keep your utility libraries small and fast. It is always worthwhile looking at any given algorithm and seeing if you can change it into a better "big-oh" category. One of the problems with a closed runtime like c55runx is that you can only guess as to the efficiency of a given library function. It's interesting to see if you could improve on, say, CLIP or INSTRING.

Now that the generic benchmarker is in place it may be a good platform for similar exercises! I would like to propose a series of challenges with the ultimate goal of building up a library of utility functions for all to use (I would be keen to see various "container" classes). To this end I would suggest that any future challenge would be permitted to use any available functionality within this library. If you have a suggestion for a future challenge, send it to editor@clarionmag.com (so he can meddle with it – ed.).

[Download the source](#)

Reader Comments

[Add a comment](#)

[I love learning. Thanks. Lets see a series. djl](#)

[Thanks Gordon for the analysis! It never occurred to me...](#)

[Interesting. You indicate that the fewer calls into the...](#)

[GeoffR, Knowing that "CASE" is slow with strings versus...](#)

[Brice Schagane, LOCAL will be insignificantly faster \(it...](#)

[Thanks for clearing that up. I was always under the...](#)

[Wow, i actually won... When i sent in my code i knew it...](#)

[My final entry missed the deadline, but is just under 20%...](#)

[I took another look at my procedure and it turns out that...](#)

[Adriaan Better still: On your NoCase logic:](#)

[Geoff you're absolutely right. While experimenting with...](#)

Copyright © 1999-2002 by [CoveComm Inc.](#) All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited.

For marketing your Applications
and Developer Accessories or to
purchase other 3rd Party Tools . . .

Developer
PLUStm

[Ads](#) · [Comments](#) · [Writers!](#) · [Privacy](#) · [Contacts](#) · [PDFs](#) · [Freebies](#) · [Open Source](#)

[Topics](#) > [Internet](#) > [Email](#)

Receiving Email With MAPI (Part 1)

by **Konrad Byers**

Published 2002-10-23

Sending outgoing email using [MAPI](#) (Messaging Application Programming Interface) has become the most widely supported and accepted way for Windows software to send email. Almost all email programs (including Outlook Express, Outlook, Lotus cc Mail, Lotus Notes, Eudora, Netscape and almost any other of any note) support the MAPI standard. In fact when the Windows operating system itself sends an email it uses MAPI. For example, if you "run" `mailto:somebody@somedomain.com` then Windows will invoke your preferred MAPI email program. Clicking a web browser email address link also brings up that MAPI email program.

MAPI email has been discussed in previous Clarion Magazine [articles](#), but for the most part these articles cover sending email. In this article I will be discussing how to *receive* email with MAPI.

What is MAPI

Before I get into the specifics of receiving email with MAPI, here's a little background.

MAPI is a multi-level architecture. At the top level are client applications, like Outlook or Outlook express (to name a few). When these client applications want to send or receive messages, they call methods in one of the available client interfaces. These include: MAPI itself; Simple MAPI, which supplies basic messaging capability; Common Messaging Calls (CMC), an X.400 version of MAPI; and the Collaboration Data Objects (CDO) library, which is an object-oriented interface.

All of these interfaces allow email clients to communicate with MAPI service providers. Service providers are typically one of three kinds: address book providers, message store

providers, and transport providers. Most of what I'll be talking about will involve a client (your Clarion application) using the MAPI calls to receive a message from a message store provider. Transport providers, incidentally, work pretty much behind the scenes – client applications don't talk to them directly, they just deliver incoming messages to the message store provider, and send any outgoing messages.

Message stores keep messages in logical entities called *folders*. A MAPI folder isn't necessarily the same thing as a Windows folder (although in theory it could be – more likely it will be a flat file, or a table in a database). In MAPI, a folder is a way of organizing messages. If you use Outlook, your Outbox and Inbox are folders, as are Sent Items and Deleted Items.

When you send an email with MAPI, your email client will first place the email into the Outbox folder. The transport provider will then take the email, send it, and leave a copy in the sent mail folder.

The beauty of MAPI is that you as a software developer do not need to know or care what kind of email server the user is actually using, or what email software they are using on their local machine. Likewise it doesn't matter to your software whether your users are connected to the Internet, or if they wish to have emails sent immediately or when they next connect, etc.; these things are the domain of email programs (and are controlled by the settings of the email program in use). Your software simply makes an operating system "send email" request with total disregard for email protocols, port numbers, login names, passwords, email servers and so forth.

This is a truly wonderful thing for developers and end users both. Developers will appreciate the almost non-existent technical support issues as compared with other email implementations, and end users will appreciate the simplicity, and the familiarity of the Sent Items and Deleted Items folders or whatever features of their particular email program they are accustomed to.

Receiving MAPI messages

Email, like the communication between developer and end-user, is a two way street. Recently I had to develop a means to process many computer-generated emails from clients. As I designed the software that generated the emails, you can probably guess they were sent using MAPI.

MAPI, though, provides a two-way interface between application software and email programs. It can be used to send outgoing emails, and just as easily to read or process *incoming* email. The reasons for using MAPI on the receiving side are the same as for using it to send. A program to process the emails does not need to know the details any more than the

sending side does. The email program I use knows what Internet connection it should use, what email account to use, and what login name and password to use; luckily it is enough for my favorite email program to know these things, so that the software I write does not need to know or care.

Receiving emails and processing them to extract information, check subjects, message bodies and attachments, etc., is made extremely easy by the class I will be discussing. Normally I would get into the nuts and bolts of how things actually get done, but because of the complexity of the class I'll just hit the highlights. The real work was writing the class to make sure it was very reusable, and extremely simple to implement. For instance, the code below is a complete program to display the subjects of any email messages currently in your MAPI email programs incoming messages queue and mark them as read (if they have not been so marked already)

```
PROGRAM
MAP
  ReadMAPICallback(MessageInfoType MessageInfo, |
    MessageAttachmentsType Attachments, <*ANY Param>),BYTE
END

INCLUDE( 'MAPI.INC' )

ReadMAPI MAPIInboxClass

CODE
  ReadMAPI.ProcessMail(ReadMAPICallback, False)

ReadMAPICallback PROCEDURE(MessageInfoType MessageInfo, |
    MessageAttachmentsType Attachments, |
    <*ANY Param>)

CODE
  STOP(MessageInfo.Subject)
  RETURN MAPIInboxCallback:Continue + MAPIInboxCallback:MarkAsRead
```

The following code shows how you can initialize the process with a little more control over things.

```
ReadMAPI.Init(True)
```

Specifying the first `Init` parameter as `True` instructs the class to download any new email messages (otherwise only the messages currently in the inbox/incoming queue may be read). The prototype for this method is:

```
Init (BYTE ReceiveMail=False, <STRING Profile>, |
    <STRING Password>, ULONG Options=0),BYTE,PROC
```

If your email program requires a profile and/or password to log in, then you may additionally specify those required values if you wish. Otherwise, the email program will prompt for

missing but required values when the request is made. (The `Options` parameter will only ever come into play if you need to optimize the email processing to be as fast as it possibly can be – I'll describe that in the `MAPIReadMail` API call section towards the end of the article.)

Believe it or not, there is only one more method, named `ProcessMail`, which you will need to call. In fact, you do not even have to call the `Init` method if you are happy with the default values as indicated by the `Init` method prototype.

The process mail prototype is:

```
ProcessMail (ProcessMailCallbackProcedure ProcessMailCallback,
            BYTE UnreadOnly=False, <*ANY Param>), BYTE, PROC
```

The `ProcessMail` method makes use of a callback procedure. This approach is quite common in Windows API (Application Programmer Interface) programming, but is not so commonly used in Clarion programming. Callbacks are typically used to implement processing where there is a joint responsibility for performing a task that is shared between you as a programmer and a *programmer interface* that implements the more general solution to the problem. It is a method that can be used to insert your own code into a generic reusable solution to a problem that is not under your own control. (For an example of callbacks, see Matt Grossmith's article [WinInet.DLL: Transferring Files With FTP \(Part3\)](#).) The `MAPIInboxClass` class implements everything necessary to read and/or process incoming email messages but does not actually process them. What kind of processing you want to do is up to you. In order to insert your own code to do so you specify a callback procedure and the "other" code calls your callback procedure that implements the portion of the processing that you are responsible for.

When the `ProcessMail` method is called it will in turn iteratively call the callback procedure for each email message. Another approach might be to build a queue of all messages, but there are drawbacks to this. You would incur all the extra overhead of building a queue of messages, and then still access the queue one item (or message) at a time anyway. Besides, maybe you only care about one message out of 10,000. If you have 10,000 messages to add to a queue, it is just an inefficient use of time and memory. The callback approach is the way to go, but your callback procedure can always build a queue if that's what you want.

The `Process` mail method is called this way:

```
ReadMAPI.ProcessMail(ReadMailCallback, False)
```

The second parameter indicates that all messages should be processed, not just those that have not yet been marked as read. A value of `True` will process only messages that are marked unread (i.e. new messages). As you can see from the method prototype, there is a third

omittable parameter. This parameter can be anything. It is not used at all by the class itself and is allowed for only because it makes it possible to convey information to the callback procedure, which you must implement to receive email message information. You could for example use the same callback procedure to process may different kinds of email, possibly with different message subjects, and control the type of processing to perform with this parameter (or what email message subject(s) to process for example). If you don't need to use it for anything just don't pass a parameter; no harm, no foul.

Now the more interesting stuff. The `ProcessMailCallbackProcedure` callback procedure is defined and/or prototyped as follows:

```
MAP
  MODULE( ' ' )
    ProcessMailCallbackProcedure(MessageInfoType MessageInfo, |
                                MessageAttachmentsType Attachments, |
                                <*ANY Param>), BYTE, TYPE
  END
END
```

An actual procedure of `ProcessMailCallbackProcedure` callback procedure type will be declared this way:

```
ReadMAPICallback PROCEDURE(MessageInfoType MessageInfo, |
                            MessageAttachmentsType Attachments, |
                            <*ANY Param>), BYTE
```

The procedure name `ReadMAPICallback` is arbitrary. It is an appropriate a name as any, given that the class declaration in this example is named `ReadMAPI`. The callback procedure need not have any particular name, *only a particular prototype*. So you just make such a procedure and specify its name (`ReadMAPICallback` in this example) as the first parameter to the `ProcessMail` method, which will take care of passing that procedure to the MAPI API. Then the callback procedure will be called once for each email message, like magic, when the `ProcessMail` method uses MAPI to read the mail.

So far, all the class does is iterate how many email messages there are by calling the callback procedure for each message. To get actual information from the email you need the `MessageInfo` and `Attachments` parameters to the callback procedure. The `MessageInfo` parameter is defined as a `MessageInfoType`, and the `Attachments` parameter is defined as a `MessageAttachmentsType`. These types are defined as follows:

```
MessageInfoType      GROUP, TYPE
OriginatorAddress    ANY   ! Senders email address
OriginatorName       ANY   ! Senders name
Subject              ANY   ! Message subject
Text                 ANY   ! Message body
Date                 LONG  ! A standard Clarion date
```

```
Time                LONG ! A standard Clarion time
                    END
```

```
MessageAttachmentsType QUEUE,TYPE
FileName            CSTRING(FILE:MaxFilePath+FILE:MaxFileName+1),AUTO
FullPathName       CSTRING(FILE:MaxFilePath+FILE:MaxFileName+1),AUTO
                    END
```

The `FileName` member of `MessageAttachmentsType` is the display name for the file attachment in the message, while the `FullPathName` member is the file name you can use to actually access the file

Here is an example of a callback procedure that displays message text where the message subject is equal to empty or blank:

```
ReadMAPICallback PROCEDURE(MessageInfoType MessageInfo, |
                        MessageAttachmentsType Attachments, |
                        <*ANY Param>),BYTE
    CODE
    IF ~ MessageInfo.Subject THEN STOP(MessageInfo.Text).
    RETURN MAPIInboxCallback:Continue
```

The following example copies email attachments to `C:\` where the message subject is equal to "Incoming Files":

```
ReadMAPICallback PROCEDURE(MessageInfoType MessageInfo, |
                        MessageAttachmentsType Attachments, |
                        <*ANY Param>),BYTE
Cntr USHORT,AUTO
    CODE
    IF MessageInfo.Subject = 'Incoming Files'
        LOOP Cntr = 1 TO RECORDS(Attachments)
            GET(Attachments, Cntr)
            COPY(Attachments.FullPathName, 'C:\')
        END
    END
    RETURN MAPIInboxCallback:Continue
```

* Notice the return value returned from these procedures. Possible return values are combinations of the following:

```
ITEMIZE(0),PRE(MAPIInboxCallback)
Done        EQUATE
Continue    EQUATE
Delete      EQUATE
MarkAsRead  EQUATE(4)
    END
```

For the examples above to not only continue to the next message but also mark the current message as read (which may send a "return receipt message" if requested when the email was

sent) the return value would be `MAPIInboxCallback:Continue` + `MAPIInboxCallback:MarkAsRead`. If you want to not only continue to the next message but also delete the current message (generally meaning to move to the deleted items or other email program dependant similar behavior), the necessary return value would be `MAPIInboxCallback:Continue` + `MAPIInboxCallback>Delete`. If the return value contains the value `MAPIInboxCallback:Done` instead of `MAPIInboxCallback:Continue` (or simply does not contain `MAPIInboxCallback:Continue`) then the callback procedure will not be called again for the next message. Sometimes you will know you found the only message you are looking for any there is no point in processing the rest of the messages.

That's all the functionality you need to retrieve email using MAPI. Next week I'll demonstrate a short program that does this, and I'll discuss the inner workings of the class.

[Download the source](#)

[Konrad Byers](#) is originally from Nova Scotia, Canada and has been living in Florida since 1995. He starting programming in Clarion with 2.1 for DOS, and is still an avid user of the Clarion C/C++ compiler. He is currently available [for hire](#). A beta version of Konrad's MAPI-enabled [eSoftAnywhere DSP & More](#) digital signal processing software for shortwave and CB users and Ham Radio operators is now available for [download](#).

Reader Comments

[Add a comment](#)

[I've posted an updated source zip from Konrad, which...](#)

Copyright © 1999-2002 by [CoveComm Inc.](#) All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited.

clarion magazine
Good help isn't that hard to find.

\$1.67 per
 issue

[Ads](#) · [Comments](#) · [Writers!](#) · [Privacy](#) · [Contacts](#) · [PDFs](#) · [Freebies](#) · [Open Source](#)

[Topics](#) > [Tips/Techniques](#) > [Clarion Language](#)

Data Structures And Algorithms Part X - Going Out Of Your Tree?

by **Alison Neal**

Published 2002-10-24

In my [previous article](#) I discussed the AVL Tree or Height Balanced Tree; in this article I will continue my discussion of the AVL Tree and show how to delete an individual item while still maintaining the sub tree heights, as well as cover some of the options available for tree traversal.

Deleting nodes

When deleting a leaf from a tree there are three possible scenarios to consider: the first is where the node to be deleted has no sub trees, the second is where the node has one sub tree, and the third is where the node has both a left and a right sub tree. The first two are reasonably easy to deal with. If there are no children then the node can just be disposed of, and if the left sub tree is null then you can delete the current node and make it equal to the right sub tree, and vice versa if the right sub tree is null. However, it becomes more complex when a node has two children, and even more complex when you also consider maintaining the tree's balance. I will now take you through an example.

Below is the code I've written:

```
AVLTree.delItem    PROCEDURE(ULONG yourVal)
RetVal    BOOL(FALSE)
CODE
  SELF.root &= SELF.Rem1(SELF.root,yourVal,RetVal)

AVLTree.Rem1 PROCEDURE(*treeNode t, ULONG yourVal,*BOOL RetVal)
temp    BOOL
CODE
  IF t &= NULL THEN return t.
```

```

IF yourVal > t.nodeVal
  temp = retVal
  t.rtree &= SELF.Rem1(t.rtree,yourVal,retVal)
IF retVal
  retval = temp
  t.balance -= 1
  CASE t.balance
    OF 0
      retVal = True
    OF -2
      IF t.ltree.balance = 1 THEN t.ltree &= SELF.LR(t.ltree).
      t &= SELF.RR(t)
      retVal = CHOOSE(t.balance,FALSE,True)
  END
END
ELSIF yourVal < t.nodeval
  temp = retVal
  t.ltree &= SELF.Rem1(t.ltree,yourVal,retVal)
IF retVal
  retVal = temp
  t.balance += 1
  CASE t.balance
    OF 0
      retVal = True
    OF 2
      IF t.rtree.balance = -1 THEN t.rtree &= SELF.RR(t.rtree).
      t &= SELF.LR(t)
      retVal = CHOOSE(t.balance,FALSE,True)
  END
END
ELSE !Found it
  SELF.Curr &= t
  IF t.ltree &= NULL
    t &= t.rtree
    DISPOSE(SELF.Curr)
    retVal = True
  ELSIF t.rtree &= NULL
    t &= t.ltree
    DISPOSE(SELF.Curr)
    retVal = True
  ELSE
    !Two sub trees
    temp = retVal
    t.ltree &= SELF.Rem2(t, t.ltree,retVal)
  IF retVal
    retVal = temp
    t.balance += 1
    CASE t.balance
      OF 0
        retVal = True
      OF 2
        IF t.rtree.balance = -1 THEN t.rtree &= SELF.RR(t.rtree).
        t &= SELF.LR(t)
        retVal = CHOOSE(t.balance,FALSE,True)
    END
  END
END

```

```

    END
  END
  RETURN t
}

AVLTree.Rem2      PROCEDURE(*treeNode t, *treeNode u, *BOOL retVal)
temp      BOOL
CODE
IF (~u.rTree &= NULL)
  temp = retVal
  u.rtree &= SELF.Rem2(t,u.rtree,retVal)
  IF retVal
    retVal = temp
    u.balance -= 1
    CASE u.balance
      OF 0
        retVal = True
        RETURN u
      OF -2
        IF ~u.rtree &= NULL
          IF u.rtree.balance = 1 THEN u.rTree &= SELF.LR(u.rtree).
        END
        u &= SELF.RR(u)
      END
    retVal = FALSE
  END
ELSE
  t.nodeVal = u.nodeVal
  SELF.Curr &= u
  u &= u.ltree
  DISPOSE(SELF.curr)
  retVal = True
  END
  RETURN u
}

AVLTree.LR      PROCEDURE(*treeNode t)
q      &treeNode
CODE
q &= t
t &= t.rtree
q.rtree &= t.ltree
t.ltree &= q
q.balance -= 1
IF t.balance > 0 THEN q.balance -= t.balance.
t.balance -= 1
IF q.balance < 0 THEN t.balance += q.balance.
RETURN t
}

AVLTree.RR      PROCEDURE(*treeNode t)
q      &treeNode
CODE
q &= t
t &= t.ltree
q.ltree &= t.rtree
t.rtree &= q
q.balance += 1
IF t.balance < 0 THEN q.balance -= t.balance.
t.balance += 1
}

```

```

IF q.balance > 0 THEN t.balance += q.balance.
RETURN t

```

Note that the LR (Left Rotate) and RR (Right Rotate) methods are exactly the same for the delete as they are for the insert. I am using exactly the same methods in both cases.

Starting with a tree that looks like figure 1, I am going to delete number 1, which is the left-most node and currently has a zero balance.

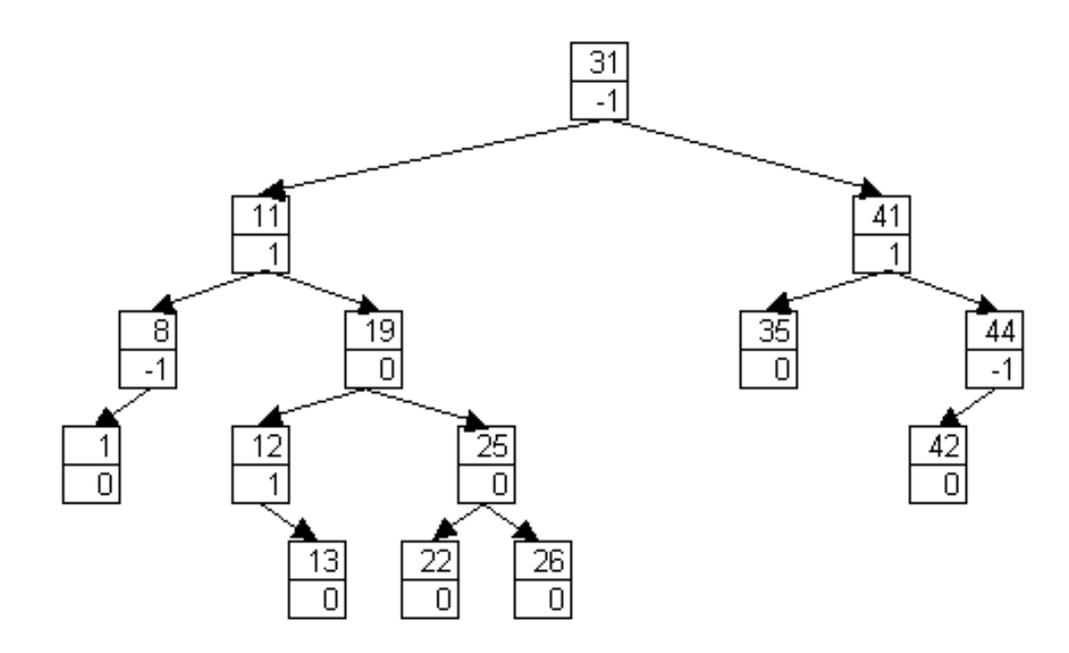


Figure 1.

First I call the `delItem` method with a parameter of the value 1. This passes the current root node (value 31) to the `Rem1` method, along with the node value of 1, and the `retVal` which is automatically set to `False`.

Now, 1 is less than 31, so the `Rem1` method is called again passing the left sub tree node containing the value 11. 1 is less than 11 so `Rem1` is called again passing the left sub tree node containing value 8. 1 is also less than 8 so again `Rem1` is called passing the next left node that contains 1, the value I'm looking for.

So, this time `SELF.Curr` is made to equal the node containing the 1, a check is made to see whether the left sub tree of 1 is null, which it is, so 1 is made to equal its right sub tree. In this case the right sub tree is also null, so now the node is no longer referred to in the tree, however if there had been a right sub tree value this would have replaced the current node.

`SELF.Curr` is then disposed of and `retVal` is made to equal `True`.

Returning back to the node containing the value 8, `retVal` is `True` so the balance of this node is adjusted to 0. Again `retVal` is set to `True` so on returning to the node containing 11 the

balance is adjusted to equal 2. Figure 2 shows the tree at this point.

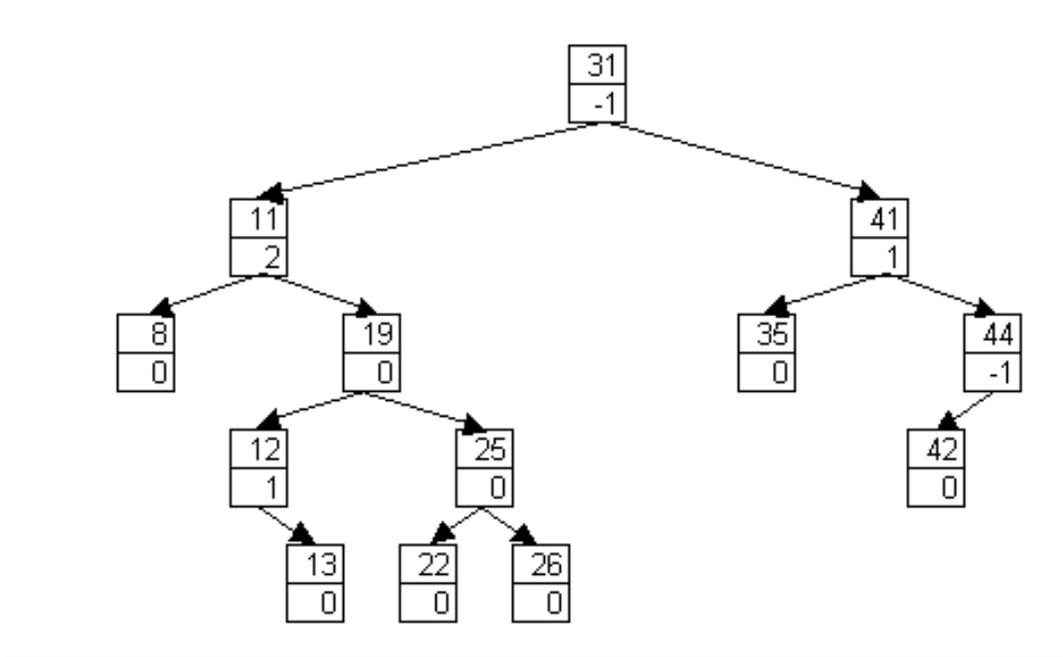


Figure 2.

Figure 2 shows that the sub trees of the node containing value 11 are now out of balance so, this has to be corrected with a Left Rotate. The LR method is called, and q is set to equal the node containing value 11, and t is set to the node containing 19. Q is then set to point right to 12, and t is set to point left to 11. The balance of 11 is adjusted to 1 and the balance of 19 is adjusted to -1 . Figure 3 shows the tree at this stage.

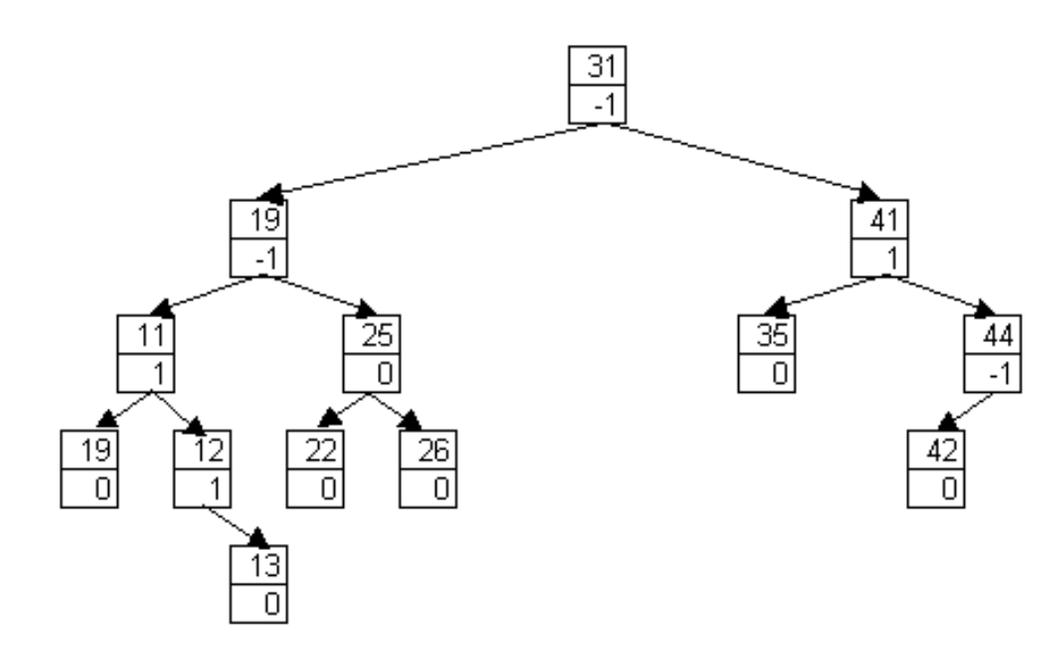


Figure 3.

On returning to Rem1, `retval` is set to `False`, so nothing happens on returning to the root node.

The next number that I want to delete is 19. So starting with the root node of 31, the program recurses left to the node containing 19. This time, however, the node has two sub trees. I need to replace the value of 19 with the value stored in the right most position of the left sub tree (value 13), as this would be the next logical number to go into the position 19 is now occupying. Once I've found this value (right most left sub tree) I can just swap the values and delete the node that contained 13, rather than having to worry about re-referencing the left and right sub trees of 19.

A call is therefore made to the Rem2 method, `t` is the node containing 19, and `u` is the node containing 11. The right sub tree of 11 is not null so a recursive call of Rem2 is made; this time `u` is 12 and `t` is still 19. Again the right sub tree of `u` is not null so another recursive call is made to Rem2, making `u` refer to 13 and `t` still 19. This time the right sub tree of 13 is null so, the value of `t` (19) is replaced with 13. Then the node containing 13 is removed. On returning to the second call of Rem2, `retval` is `True`, so the balance of the node containing 12 is updated to 0. Returning to the first call of Rem2, `retval` is also `True` so the balance of the node containing 11 is updated to 0.

The program then returns to Rem1 and `retval` is `True` so the balance of the node containing 13 is also set to 0. Returning once more to the root node containing 31, `retval` is still `True` so the balance is updated once more to 0. The tree now looks like Figure 4.

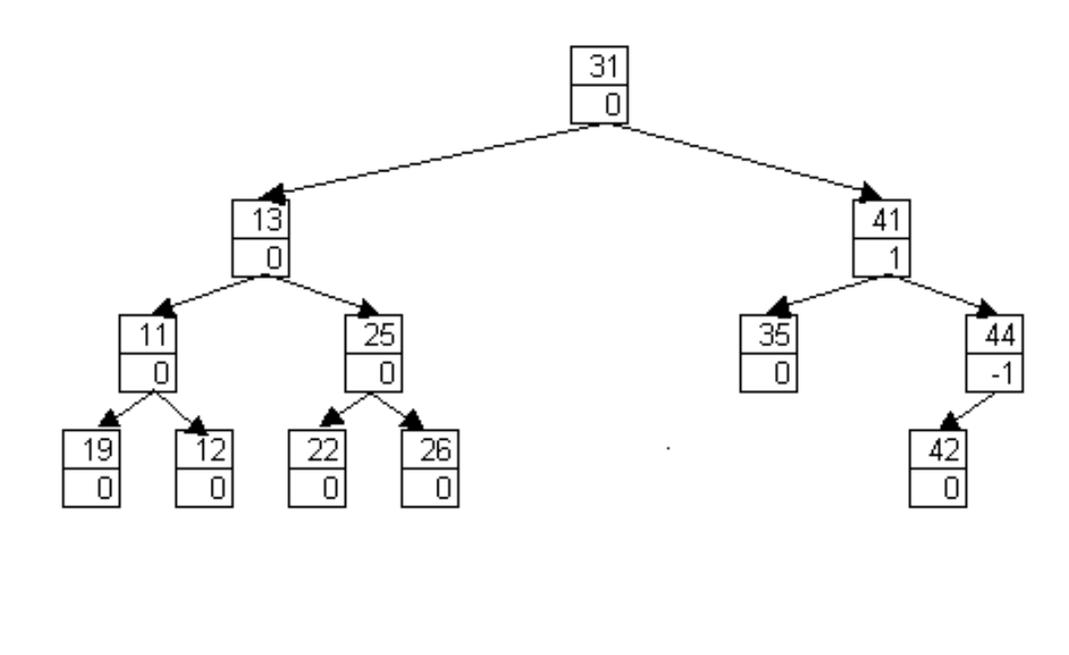


Figure 4.

What will happen should I wish to delete a node with no right-most left sub tree? If a node has

a left sub tree then it must also have a right most left sub node; even if there is only one node in the left sub tree then that is the value to be swapped. So now I want to look at deleting value 41.

Calling `Rem1` with a value of 41 and the root node (31), the method recurses right, to the node containing 41. As this is the value I want to delete and it has both a right and a left sub tree, a call is made to `Rem2`. The `t` variable refers to the node containing 41, and `u` refers to the node containing 35. This time 35 does not have a right sub tree, so the value (35) is copied into the node containing 41, and the node is deleted.

On returning to `Rem1`, `retVal` is `True` so the balance of the node (now) containing 35 is updated to equal 2. A right rotate (RR) is then performed on the nodes containing 44 and 42, so that 42 now points down right to 44, and a left rotate (LR) is performed on 44. Then the balances are all updated appropriately, giving Figure 5.

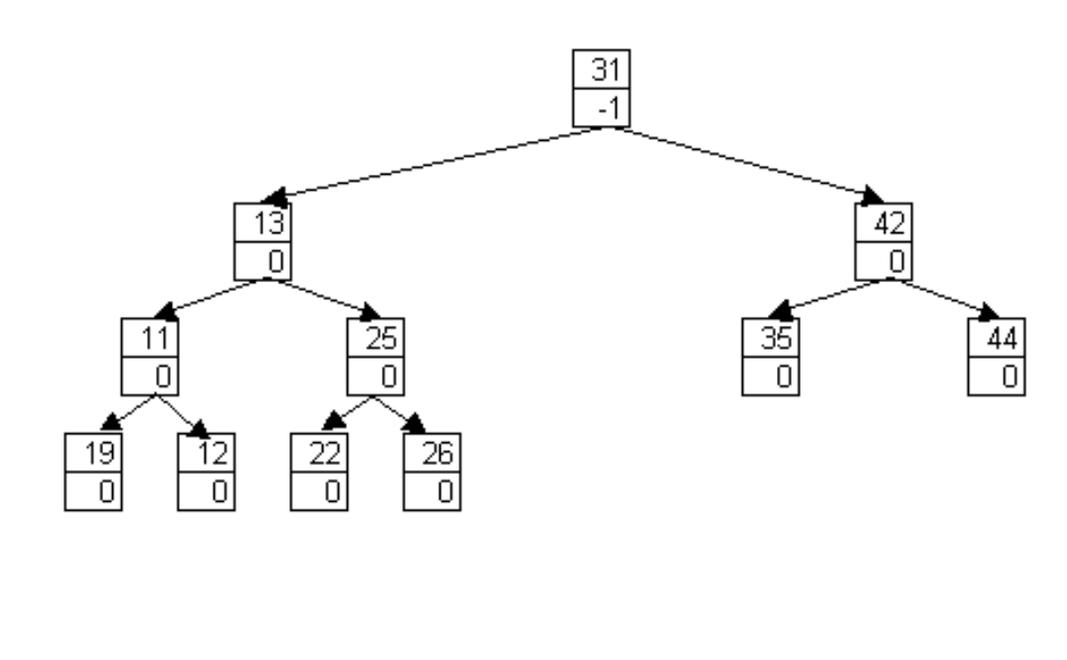


Figure 5.

Tree traversal

There are three generally accepted methods of traversing a tree. These are known as pre-order, in-order, and post-order traversal algorithms. The simplest of these is the in-order traversal. This method requires reading the tree from the left-most node to the right-most node, going from the lowest value position to the highest value position.

```

AVLTree.AVLInOrder          PROCEDURE(*treeNode t)
CODE
IF ~t &= NULL
    SELF.AVLInorder(t.ltree)
  
```

```

    EXP:MessageLine = CLIP(EXP:MessageLine) &t.nodeVal &', '
    SELF.AVLInorder(t.rtree)
END

```

The code above shows the in-order traversal, and it is basically the same algorithm for loading a perfectly balanced tree with data that has already been sorted. Pre-order traversal requires you to visit and process the root node and then to recursively visit all the nodes in the left sub tree, followed by all nodes in the right sub tree. The pre-order traversal is given below:

```

AVLTree.AVLPreOrder          PROCEDURE(*treeNode t)
CODE
IF ~t &= NULL
    EXP:MessageLine = CLIP(EXP:MessageLine) &t.nodeVal &', '
    SELF.AVLInorder(t.ltree)
    SELF.AVLInorder(t.rtree)
END

```

The Post-order traversal requires you to visit all the nodes in the left sub tree, then the nodes in the right sub tree, and then the root node:

```

AVLTree.AVLPostOrder        PROCEDURE(*treeNode t)
CODE
IF ~t &= NULL
    SELF.AVLInorder(t.ltree)
    SELF.AVLInorder(t.rtree)
    EXP:MessageLine = CLIP(EXP:MessageLine) &t.nodeVal &', '
END

```

It is also possible to perform the in-order traversal using iteration rather than recursion – this is known as the Morris In-Order Algorithm:

```

AVLTree.MorrisInOrder       PROCEDURE()
t &treeNode
p &treeNode
CODE
CLEAR(EXP:MessageLine)
p &= SELF.root
LOOP WHILE ~p &= NULL
    IF p.ltree &= NULL
        EXP:MessageLine = CLIP(EXP:MessageLine) &p.nodeVal &', '
        p &= p.rtree
    ELSE
        t &= p.ltree
        LOOP WHILE ~t.rtree &= NULL AND ~t.rtree &= p
            t &= t.rtree
        END
        IF t.rtree &= NULL
            t.rtree &= p
            p &= p.ltree
        ELSE
            EXP:MessageLine = CLIP(EXP:MessageLine) &p.nodeVal &', '
            t.rtree &= NULL

```

```
        p &= p.rtree
    END
END
END
ADD(ExportFile)
```

Summary

The Binary Search tree can be far more efficient than a linked list when you have to search for items on a frequent basis. However, unless the tree is weight balanced or height balanced, it runs the risk of collapsing into a linked list. Unfortunately frequent additions and deletions to the balanced trees come with large overheads, as the nodes have to be rotated to maintain that much-desired balance. Tree and tree algorithms though have their uses, and the overheads are reduced by the fact that you're mainly only reassigning reference variables and not values stored in memory.

[Download the source](#)

Alison Neal has been using Clarion since 2000, whilst working for [Asset Information Systems](#) (AIS) in Auckland, New Zealand. Some years ago (at the tender age of 19) Alison graduated from the Central Institute of Technology in Wellington, New Zealand with a major in Cobol. She also has a BA in English literature and has studied Computer Science, Philosophy and Information Systems. AIS is an independent division of Asset Forestry Ltd, and has a team of five programmers developing almost exclusively in Clarion. AIS also offers web (ClarionNET) and email services for the customer who needs everything. The company has many and varied customers bridging across a wide range of industries including Telecommunications, Forestry & Agriculture, Manufacturers, Military & Government, Legal & Financial, and Retail.

Reader Comments

[Add a comment](#)

Copyright © 1999-2002 by [CoveComm Inc.](#) All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited.

INVEST
in your own abilities**Clarion**
magazine[Ads](#) · [Comments](#) · [Writers!](#) · [Privacy](#) · [Contacts](#) · [PDFs](#) · [Freebies](#) · [Open Source](#)[Topics](#) > [Tips/Techniques](#) > [Clarion Challenge](#)

ContainsMatch A to Z

by **Phil Will**

Published 2002-10-24

I really enjoyed the recent [ClarionMag ContainsMatch challenge](#). As I got deeper into it, though, I had a nagging feeling that the ContainsMatch function really ought to be internationalized. Since 1994, when Clarion for Windows first came out, internationalization of code has always seemed to me to be treated as an afterthought by Clarion developers. Even though the specifications for an application may not include internationalization, it is often easy to do things that will make it "global ready" should the need arise. So here's a modified version of my submission (the one that failed to get there on time and which I'm sure would have won...).

The challenge

The challenge specification called for determining if characters in a string could be found in a second string. The match could be either case sensitive or not case sensitive. The space character was to be ignored. The non-internationalized part was that only the characters 0-9, a-z, and A-Z were to be considered – "A-Z" thinking that ignores most of the world's alphabetic characters. To internationalize the spec, the function must handle all alpha characters in the extended character sets. The procedure and its prototype are still as follows.

```
ContainsMatch PROCEDURE(string txt, string chars, byte noCase=false), byte
```

Late submission

Listing 1 below shows my late submission *before* internationalization. It has two main characteristics: 1) it uses several 255 element byte arrays to do most of the work, and 2) it is datacentric. The elements in the arrays are either 1 or 0 and the subscript corresponds to the ASCII value of a character.

There are three key pairs of data, each consisting of a 255 character string and a 255 element array that is declared over the string. The AlphaArray has elements that correspond to 0-9, a-z, and A-Z; if the array subscript matches the numeric value of one of these characters, then the ASCII value for that array element is set to 1, and if not it is set to 0. The CharIsUpper Array has only those elements that correspond to the upper alpha characters set to true. The CharIsLower Array does the same thing with lower case characters. Note that consecutive occurrences of the same character within a string constant may be represented by curly brace repeat count notation, i.e. (<1>{26}) means that the value 1 is repeated 26 times.

Listing 1

```
ContainsMatch PROCEDURE(string txt,string chars,|
                                byte noCase=false) !, byte
I          SHORT,AUTO
TxtArray  BYTE,DIM(256),AUTO
Found     BYTE,AUTO
ThisChar  BYTE,AUTO
eLower    EQUATE(32) ! Difference between upper and lower VALs
          ! 1=IsAlphaNumeric 0-9 a-z A-Z
Alpha     STRING('<0>{47}<1>{10}<0>{7}<1>{26}<0>{6}<1>{26}<0>{133}')
AlphaArray BYTE,DIM(255),OVER(ALPHA)
          ! 1=IsUpper A-Z
UpperStr  STRING('<0>{64}<1>{26}<0>{165}')
CharIsUpper BYTE,DIM(255),OVER(UpperStr)
          ! 1=IsLower a-z
LowerStr  STRING('<0>{96}<1>{26}<0>{133}')
CharIsLower BYTE,DIM(255),OVER(LowerStr)
CODE
! Upper if not case sensitive.
CLEAR(TxtArray)
! Load Text Array (ignores 0)
LOOP I=1 TO LEN(Txt)
  TxtArray[Val(Txt[I])]=TRUE
END
! Check VAL
Found=1
LOOP I=1 TO LEN(Chars)
  ThisChar=VAL(Chars[I])
  IF AlphaArray[ThisChar]
    IF TxtArray[ThisChar] THEN CYCLE.
    IF NoCase
      IF CharIsLower[ThisChar]
        IF TxtArray[ThisChar-eLower] THEN CYCLE.
      ELSIF CharIsUpper[ThisChar]
        IF TxtArray[ThisChar+eLower] THEN CYCLE.
    END
  END
  Found=FALSE
  BREAK
END
END
RETURN FOUND
```

The ContainsMatch procedure declares an additional 255 byte array. The code sets all matching array elements to TRUE by looping through the target txt string characters (the first LOOP in the listing). The second LOOP then loops through each character in the Chars search string to see if it can be found. First its ASCII value is found and assigned to ThisChar using the VAL function. The statement IF AlphaArray[ThisChar] returns True if it is an alpha character. TxtArray[ThisChar] returns True if there is an exact match in the text to be searched. If there's no match and the comparison is case insensitive (NoCase is true), the code CharIsLower[ThisChar] determines if it is lower case. The remaining code changes the TxtArray index to upper or lower case values by adding or subtracting the difference between upper and lower case ascii values. If the values match, the loop cycles. If not, it changes the Found return value to false and exits the loop.

International rethink

This code above can easily be internationalized without much loss in performance. The trick is to change the data centric strings and arrays to international versions that can be set at run time. Listing 2 does this using a class with two methods. The first method sets the array elements. I've use four standard Clarion functions to do this: ISALPHA, ISUPPER, ISLOWER, and NUMERIC. The only substantive change to the ContainsMatch procedure is to use the VAL and UPPER functions to do case insensitive comparisons. These upper or lower the character and then get its ASCII value: VAL(UPPER(pChars[I]))

Listing 2

```

CM          CLASS
           ! 1=IsAlphaNumeric 0-9 a-z A-Z
AlphaStr    STRING(' <0>{47}<1>{10}<0>{7}<1>{26}<0>{6}<1>{26}<0>{133}')
AlphaArray  BYTE, DIM(255), OVER(ALPHASTR)
           ! 1=IsUpper A-Z
UpperStr    STRING(' <0>{64}<1>{26}<0>{165}')
CharIsUpper BYTE, DIM(255), OVER(UpperStr)
           ! 1=IsLower a-z
LowerStr    STRING(' <0>{96}<1>{26}<0>{133}')
CharIsLower BYTE, DIM(255), OVER(LowerStr)
Construct   PROCEDURE
ContainsMatch PROCEDURE(string txt, string chars, byte noCase=false), byte
           END

CM.Init PROCEDURE
I  BYTE, AUTO
  CODE
  LOOP I=1 TO 255
    IF NUMERIC(CHR(I))
      SELF.AlphaStr[I]=TRUE
    ELSIF CHR(I)=' ' ! Space
    ELSE
      SELF.AlphaStr[I]=ISALPHA(CHR(I))

```

```

        SELF.UpperStr[I]=ISUPPER(CHR(I))
        SELF.LowerStr[I]=ISLOWER(CHR(I))
    END
END

CM.ContainsMatch PROCEDURE(STRING pTxt,STRING pChars, |
                        BYTE pNoCase=false) !, byte
I          SHORT,AUTO
TxtArray  BYTE,DIM(256),AUTO
Found     BYTE,AUTO
ThisChar   BYTE,AUTO
CODE
CLEAR(TxtArray)
! Load Text Array, setting to true if character found.
LOOP I=1 TO LEN(CLIP(Txt))
    pTxtArray[Val(Txt[I])]=TRUE
END
! Check VAL
Found=1
LOOP I=1 TO LEN(CLIP(Chars))
    ThisChar=VAL(pChars[I])
    IF SELF.AlphaArray[ThisChar]
        IF TxtArray[ThisChar] THEN CYCLE.
    IF NoCase
        IF SELF.CharIsLower[ThisChar]
            IF TxtArray[VAL(UPPER(pChars[I]))] THEN CYCLE.
        ELSIF SELF.CharIsUpper[ThisChar]
            IF TxtArray[VAL(LOWER(pChars[I]))] THEN CYCLE.
        END
    END
    Found=FALSE
    BREAK
END
END
RETURN FOUND

```

It doesn't work!

I assume at this point that you have copied the code, put it into a test application, tried it, and found it doesn't work! The secret is that those added functions, ISALPHA, ISUPPER, ISLOWER, UPPER, and LOWER, will not recognize extended characters unless the Clarion CLACASE Environment has been properly set. Listing 3 shows a short test program that does this. It uses the LOCALE function to set the CLACASE value, calls the Init method, and then does two comparison about 20000 times. That test takes a little over one second on my machine.

Listing 3

```

PROGRAM
MAP.
FrUpper  EQUATE('ABCDEFGHIJKLMNOPQRSTUVWXYZfšĚŽšÀÁÂÃÄÅÆÇÈÉÊËÌÍÎÏÐÑÒÓÔÕÖÙÚÛÜÝÞÿ')
FrLower  EQUATE('abcdefghijklmnopqrstuvwxyzfšæžšÀÁÂÃÄÅÆÇÈÉÊËÌÍÎÏÐÑÒÓÔÕÖÙÚÛÜÝÞÿ')

```

```
EnUpper EQUATE( 'ABCDEFGHIJKLMNOPQRSTUVWXYZ' )
EnLower EQUATE( 'abcdefghijklmnopqrstuvwxyz' )
clock1 LONG
rvNoCase BYTE, AUTO
rvCase BYTE, AUTO
eNoCase EQUATE(1)
eCase EQUATE(0)
CODE
LOCALE( 'CLACASE', FrUpper&', '&FrLower )
CM.Init
clock1=clock()
LOOP 200000 TIMES
    rvCase=CM.ContainsMatch( frUpper&frLower, frUpper&frLower, eCase )
    rvNoCase=CM.ContainsMatch( frUpper&frLower, frUpper&frLower, eNoCase )
END
stop(clock()-clock1&' Case: '&rvCase&' No Case: '&rvNoCase)
```

Other submissions to the challenge could probably handle international environments with minor modification. Hopefully, next time you start to code using anything like `CASE MyString[I]; OF 'A' TO 'Z' OROF 'a' TO 'z'` you will have the same nagging thought I did, and will consider what steps you might need to take to add international support.

[Philip S. Will](#) is President of [ProDomus, Inc.](#), a SoftVelocity Third Party Accessories Partner and Clarion applications developer. He has been a presenter at several Clarion conferences, and has published articles on Internationalization and template writing. His principal third party products include PD Lookups, PD Translator Plus, and PD 1-Touch Date Tools. Philip has been coding in Clarion since 1991.

Reader Comments

[Add a comment](#)

Reborn Free**CLARION**
online[Ads](#) · [Comments](#) · [Writers!](#) · [Privacy](#) · [Contacts](#) · [PDFs](#) · [Freebies](#) · [Open Source](#)[Topics](#) > [Internet](#) > [Email](#)

Receiving Email With MAPI (Part 2)

by Konrad Byers

Published 2002-10-30

Last week I discussed the basics of retrieving email using MAPI. Now it's time to put all this code together into a working example. The following is a simple program which marks incoming messages as read and displays the number of file attachments:

PROGRAM

MAP

```
ReadMAPICallback(MessageInfoType MessageInfo, |
                  MessageAttachmentsType Attachments, |
                  <*ANY Param>),BYTE
```

END

INCLUDE('MAPI.INC')

ReadMAPI MAPIInboxClass

CODE

```
ReadMAPI.Init(False)
ReadMAPI.ProcessMail(ReadMAPICallback, False)
```

```
ReadMAPICallback PROCEDURE(MessageInfoType MessageInfo, |
                             MessageAttachmentsType Attachments, |
                             <*ANY Param>)
```

CODE

```
STOP(MessageInfo.Subject & '- ' |
      & CLIP(LEFT(RECORDS(Attachments))) & ' Attachments')
RETURN MAPIInboxCallback:Continue + |
      MAPIInboxCallback:MarkAsRead
```

Sending mail

The class source code also includes a class to send email. For now, in a nutshell, the methods

of the email sending class you will use will be as follows:

- `SetSubject PROCEDURE (STRING Subject)`
- `SetMessage PROCEDURE (STRING MessageText)`
- `SetToName PROCEDURE (STRING ToName)` - **Used only to "name" a group of recipients added with the following methods (so that it will appear to be to this "group name" rather than many individuals; "all employees" for example)**
- `AddTo PROCEDURE (STRING Addr)` - **Add just an email address recipient**
- `AddTo PROCEDURE (STRING Name, STRING Addr)` - **Add a recipient with name & email**
- `AddCc PROCEDURE (STRING Addr)` - **Add CC (carbon copy) with email address only**
- `AddCc PROCEDURE (STRING Name, STRING Addr)` - **Add CC (carbon copy) with name & email address**
- `AddBcc PROCEDURE (STRING Addr)` - **Add BCC (blind carbon copy) with email address only**
- `AddBcc PROCEDURE (STRING Name, STRING Addr)` - **Add BCC (blind carbon copy) with name & email address**
- `AddAttachment PROCEDURE (STRING FileName, BYTE UseACopyOfTheFile=False)`

And finally,

- `SendMail PROCEDURE (BYTE Dialog=False), ULONG, PROC` - `Dialog=True` **will cause an email window to appear for user interaction**

Calling methods to set properties of an outgoing email (such as recipients, attachments, etc.) creates an internal state that will be invoked with the `SendMail` method. If the `SendMail` method is called twice in a row (with the same parameter), two identical operating system send requests will occur (using the same internal state for recipients, attachments, etc.). If another recipient is added and the `SendMail` method is called again, an identical email send request is made but with an *additional* recipient because the previous internal state is preserved. To send a completely different email you reset the class's internal state with the `Reset` method; then you can compose a new email.

Because "Clear" methods exist for resetting only particular properties rather than the entire internal state, this preservation of internal state can be a great convenience when sending multiple emails. You can construct a complicated *email state*, and then use the same state to send many emails with only slight variations (to may different recipients for example).

This class uses only one MAPI API call but it is a rather complicated one (with pointers to memory addresses to pointers to memory addresses to pointers to memory addresses). The class itself though is still very easy to use.

This is a simple example program to send an email:

```
PROGRAM
MAP
END

INCLUDE ( 'MAPI.INC' )

SendMAPI MAPIMailClass

CODE
SendMAPI.SendMail ( True )
```

The above does not do much because nothing much was specified by way of the class's methods, but because the `Dialog` parameter of `SendMail` is `True` an email-sending *dialog window* will be displayed. The dialog window in this context refers to the send email entry form of whatever MAPI email program is in use. (The dialog window form must be accepted or cancelled by the user in the normal fashion for a manually sent email.)

This is the simplest form of sending a MAPI email; having no recipient(s), subject, attachments or message text specified. You can call additional class methods to specify such things as recipients, message text and etc. In this case, with additional information specified and still using the `Dialog` parameter option as `True` will provide the user with the same "send email" form, but with the specified information already filed in. Clicking on an email link in your web browser is an example of this (i.e. the recipient email address is filed in and additional information is specified manually by the user and the email is accepted/sent by the user). Given that all the necessary information is specified using the appropriate class methods, and is correct, the `Dialog` parameter of the `SendMail` method may then be specified as `False` causing the email to be sent without any user interaction (no form to be cancelled or accepted is displayed it is just accepted).

If there is enough interest I will gladly write another article on sending MAPI email using this class. Just to tweak your interest in this class, faxes can also be sent (which is also *sort of* included in the MAPI standard).

MAPI receive calls

The receiving side code uses six MAPI Windows API calls. Namely these procedures are `MAPILogon`, `MAPIFindNext`, `MAPIReadMail`, `MAPIFreeBuffer`, `MAPIDeleteMail`, and `MAPILogoff`. The complete prototypes, straight from the API documentation, are listed below. (When it comes to Windows API documentation it is a C programmers world.)

```

ULONG FAR PASCAL MAPILogon(
    ULONG ulUIParam,
    LPTSTR lpszProfileName,
    LPTSTR lpszPassword,
    FLAGS flFlags,
    ULONG ulReserved,
    LPLHANDLE lplhSession
)

ULONG FAR PASCAL MAPIReadMail(
    LHANDLE lhSession,
    ULONG ulUIParam,
    LPTSTR lpszMessageID,
    FLAGS flFlags,
    ULONG ulReserved,
    lpMapiMessage FAR * lppMessage
)

ULONG MAPIFreeBuffer(
    LPVOID lpBuffer
)

ULONG FAR PASCAL MAPIDeleteMail(
    LHANDLE lhSession,
    ULONG ulUIParam,
    LPTSTR lpszMessageID,
    FLAGS flFlags,
    ULONG ulReserved
)

ULONG FAR PASCAL MAPILogoff (
    LHANDLE lhSession,
    ULONG ulUIParam,
    FLAGS flFlags,
    ULONG ulReserved
)

```

As you can probably guess, of the procedures above `MAPILogon` and `MAPILogoff` will be the first and last calls to the MAPI interface. Logging on with `MAPILogon` is performed by the class using the settings specified with the `Init` method (and the defaults otherwise). You can specify a profile name and password, and indicate whether or not you want to download email or not as determined by the `Init` method, which controls the value of the `dwFlags` parameter. The `dwFlags` parameter will otherwise be composed of the equated API values of `MAPI_NEW_SESSION` and `MAPI_LOGON_UI`. These values indicate respectively: 1) an attempt should be made to create a new session rather than acquire the environment's shared session; and 2) a logon dialog box should be displayed to prompt the user for logon information (if the user needs to provide a missing password and profile name to enable a successful logon). `MAPILogoff` receives the value that `MAPILogon` obtains referenced by the `lplhSession` parameter, and likewise all the other API calls must receive this value.

`MAPIReadMail` receives the session handle from `MAPILogon` and receives a message ID

(that is implemented as a sort of incrementing message counter by the class wrapper), a reference variable to what will end up as the `MessageInfo` parameter to your callback procedure, and some important flags that are determined by the `Init` method and the return value from your callback procedure. The API equate values are: 1) `MAPI_BODY_AS_FILE - MAPIReadMail` should write the message text to a temporary file and add it as the first attachment in the attachment list; 2) `MAPI_ENVELOPE_ONLY - MAPIReadMail` should read the message header only. File attachments are not copied to temporary files, and neither temporary file names nor message text is written for the purpose of enhancing performance when neither attachments nor message body are required; 3) `MAPI_PEEK - MAPIReadMail` does not mark the message as read which can affect its appearance in the user interface and generate a read receipt; 4) `MAPI_SUPPRESS_ATTACH - MAPIReadMail` should not copy file attachments but should write message text into the `MapiMessage` structure for the purpose of enhancing performance when attachments are not required. The `MAPI_PEEK` value is controlled by the return value from your callback while all others may be specified with the `Options` parameter of the `Init` method.

When the memory allocated by MAPI for the `MapiMessage` structure used with `MAPIReadMail` is no longer required, the `MAPIFreeBuffer` procedure must be called to release this memory. `MAPIFreeBuffer` receives only a reference to the structure to free.

The class wrapper calls the `MAPIDeleteMail` procedure when your callback procedure returns a value containing `MAPIInboxCallback:Delete`. `MAPIDeleteMail` receives the session ID (obtained with `MAPILogon`) and the message ID and will cause the message to be removed from the incoming mail queue (i.e. deleted).

The LoadDLL Class

Both MAPI class wrappers (for send and receive) use the `LoadDLL` class (also included). This is because MAPI is an optional Windows component. If you used the normal Clarion approach of making a LIB with the `LibMaker.exe` utility and linking the LIB file (adding it to the project settings), your programs would refuse to run when `MAPI32.DLL` could not be loaded. If the DLL was guaranteed to exist on every machine I would have just used a LIB, but the `LoadDLL` class is useful here because unlike using a LIB it allows a program to load a DLL *when it can*.

`LoadDLL` allows programmers to report or handle the error any way they wish, but it will not automatically cause an immediate program halt when the program is initializing itself if an error loading the DLL occurs (unlike using a LIB). This class will hopefully be the subject of a future article. For now I will say that it has the nice feature of being a finishing touch rather than a part of the development. I did my development and initial testing of these classes using a LIB file for `MAPI32.DLL`. When it was working I removed the LIB and added in a few lines

of code to use the LoadDLL class instead while the rest of the code is still as it was when I used a LIB. It is different in this respect from the [class Jim Kane created](#), which I incidentally use as well.

Summary

MAPI is a vast and complicated subject. I have covered the most basic elements of what Microsoft refers to as the Simple MAPI API. (You don't even want to know what Microsoft calls complicated these days). There are only a few capabilities of Simple MAPI that I didn't cover, including ambiguous name and address resolution (optionally with or without user intervention), address book browsing and general maintenance, and dialogs for addressing messages that are essentially the same as that for filling out a complete message but with the ability to add recipients only (no attachments, message text, etc).

What you have in these classes is the ability to send outgoing email and receive incoming email using MAPI that can replace existing POP and SMTP email processing. I am sure that you will someday find yourself, in hindsight, grateful to be using MAPI, just as us old-timers are grateful to be using Windows printer drivers rather than control codes for a dozen different kinds of printers. Standard APIs make our lives easier, and our users' lives easier, and that's good for business.

[Download the source](#)

[Konrad Byers](#) is originally from Nova Scotia, Canada and has been living in Florida since 1995. He starting programming in Clarion with 2.1 for DOS, and is still an avid user of the Clarion C/C++ compiler. He is currently available [for hire](#). A beta version of Konrad's MAPI-enabled [eSoftAnywhere DSP & More](#) digital signal processing software for shortwave and CB users and Ham Radio operators is now available for [download](#).

Reader Comments

[Add a comment](#)

Copyright © 1999-2002 by [CoveComm Inc.](#) All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited.

Reborn Free**CLARION**
online[Ads](#) · [Comments](#) · [Writers!](#) · [Privacy](#) · [Contacts](#) · [PDFs](#) · [Freebies](#) · [Open Source](#)[Topics](#) > [Databases](#) > [SQL](#)

Converting The Inventory Example - Calling Stored Procedures (Part One)

by Ayo Ogundahunsi

Published 2002-10-31

In my [previous articles](#), I showed how to connect to SQL Server, import data, and generally run a Clarion application with a SQL Server backend. In this series of articles, I will show you how to call a stored procedure, pass a parameter, and (hopefully) get results from the stored procedure. I will also provide a template to make this easy to do.

The first step is to split the inventory application to an EXE and DLL(s). While this is not a specific requirement, it is better to [organize your applications this way](#). Depending on how big an application is, you may wish put all your reporting procedures in one DLL, all your browses and forms in another DLL, etc. A common practice is to have a DLL just for data elements, i.e. file declarations and other global data. The EXE and other DLLs still have those declarations, but everywhere except in the data DLL they are declared as EXTERNAL - the memory is only allocated in the data DLL.

For the inventory application, I will be calling the data DLL INV_SQL_DATA.DLL. For most part, this article will focus on the process of splitting the inventory app into DLLs.

Data access and update layers

One good reason to have a data DLL is that I want to have only one access point for all calls I make to SQL Server. In other words, I don't want to have embedded SQL all over my applications; I only want one point of maintenance, and the less embedded SQL code I have in my application, the easier it is to have the business logic external and available for non-Clarion use. Figure 1 demonstrates the data access logic.

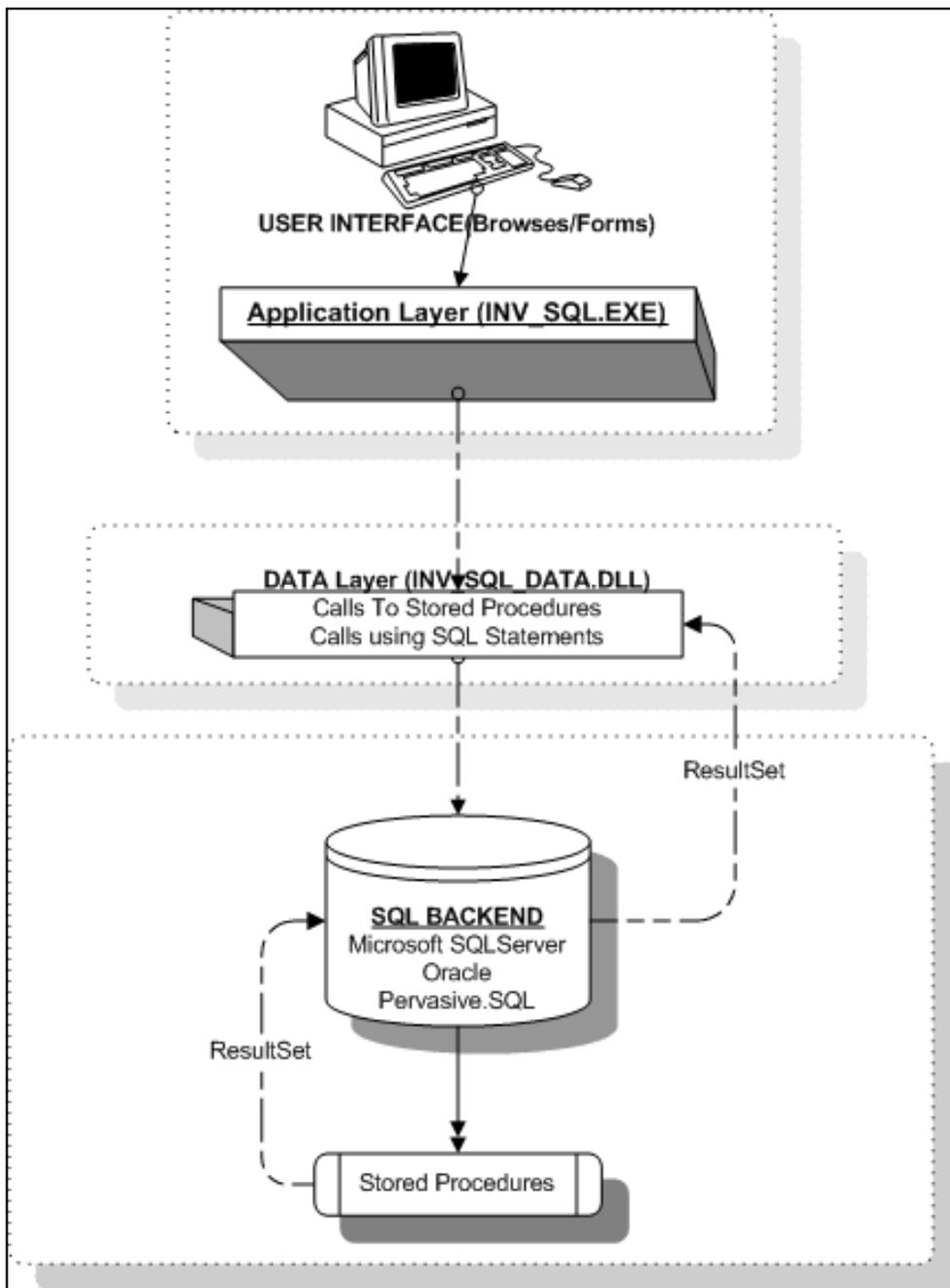


Figure 1. Data Access Logic

Application Layer – This layer will contain the call to the procedure (located in INV_SQL_DATA.DLL) that interacts with SQL Server. For example, the procedure *ProcessPrices* uses the *Process* template to change the prices for all the products in the Inventory. To make the application more SQL-like, I will replace this process with a stored procedure. Therefore, instead of making the call to the stored procedure inside INV_SQL.APP, I will make the call inside INV_SQL_DATA.APP.

Data Layer – This layer will contain the table definitions, all calls to stored procedures, all SQL statements that will be executed, and a connector procedure which replaces the connection string initialization (e.g. `GLO:ConnectionString= '(LOCAL),inv_sql,sa,sa'`) which you would otherwise

put in an embed point.

Creating a DLL

Step 1: Create a new application as a DLL, as shown in Figure 2.

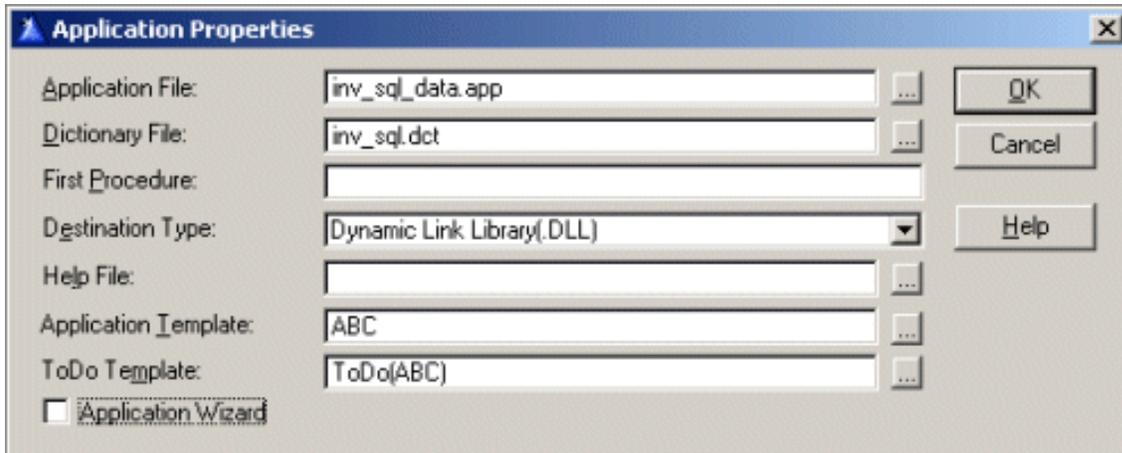


Figure 2. Creating a DLL - Application Properties

Step 2: Make sure the project settings are correct, as shown in Figure 3.

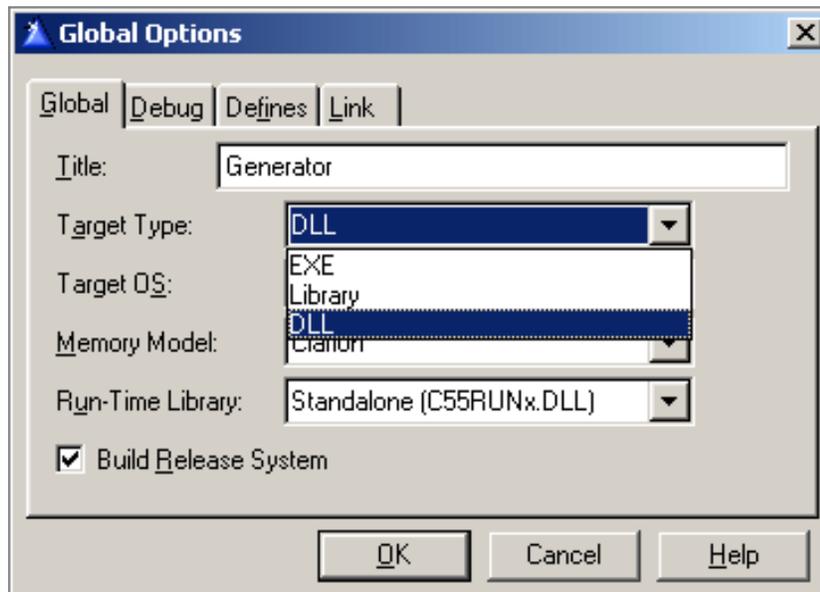


Figure 3. Creating a DLL – Global Options

Step 3: Set Global Properties – General. Uncheck the Generate template globals and ABC's as EXTERNAL option. See the Clarion Help for more explanation on this.

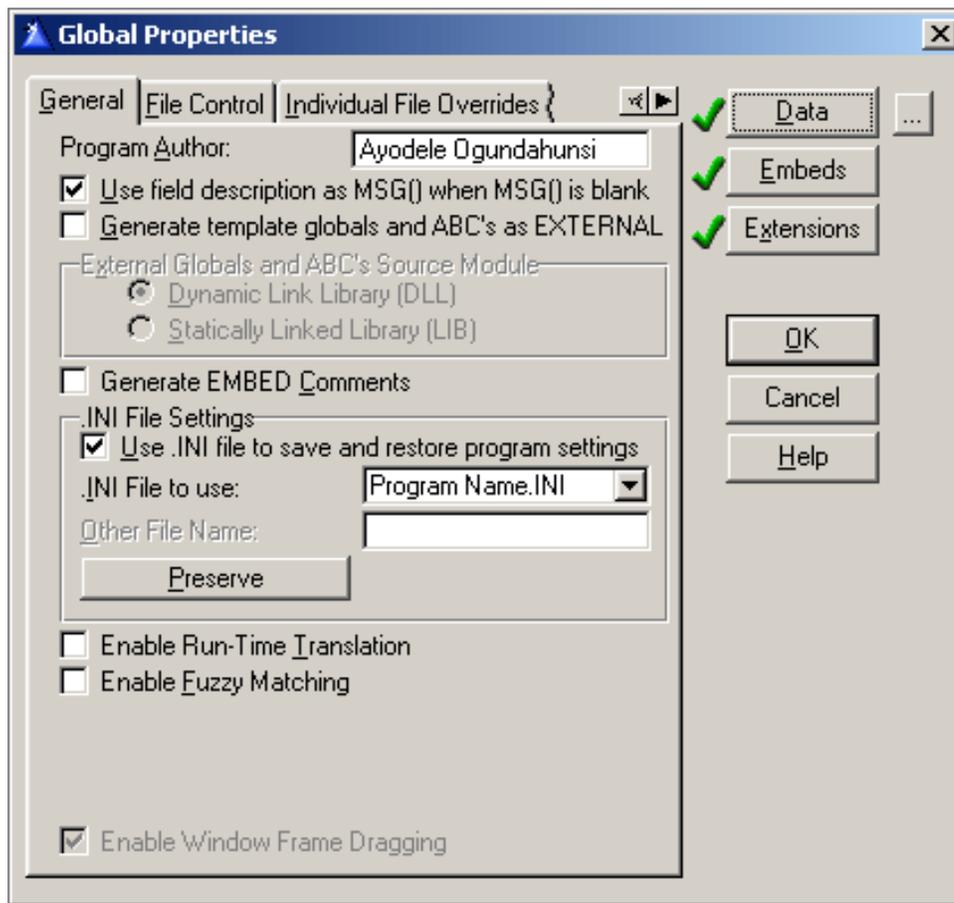


Figure 4. Setting Global Properties - General

Step 4: Set Global Properties – File Control. Check Generate all file declarations, Change the File Attributes (External) to None External, and check Export all file declarations, and uncheck Defer opening files until accessed.

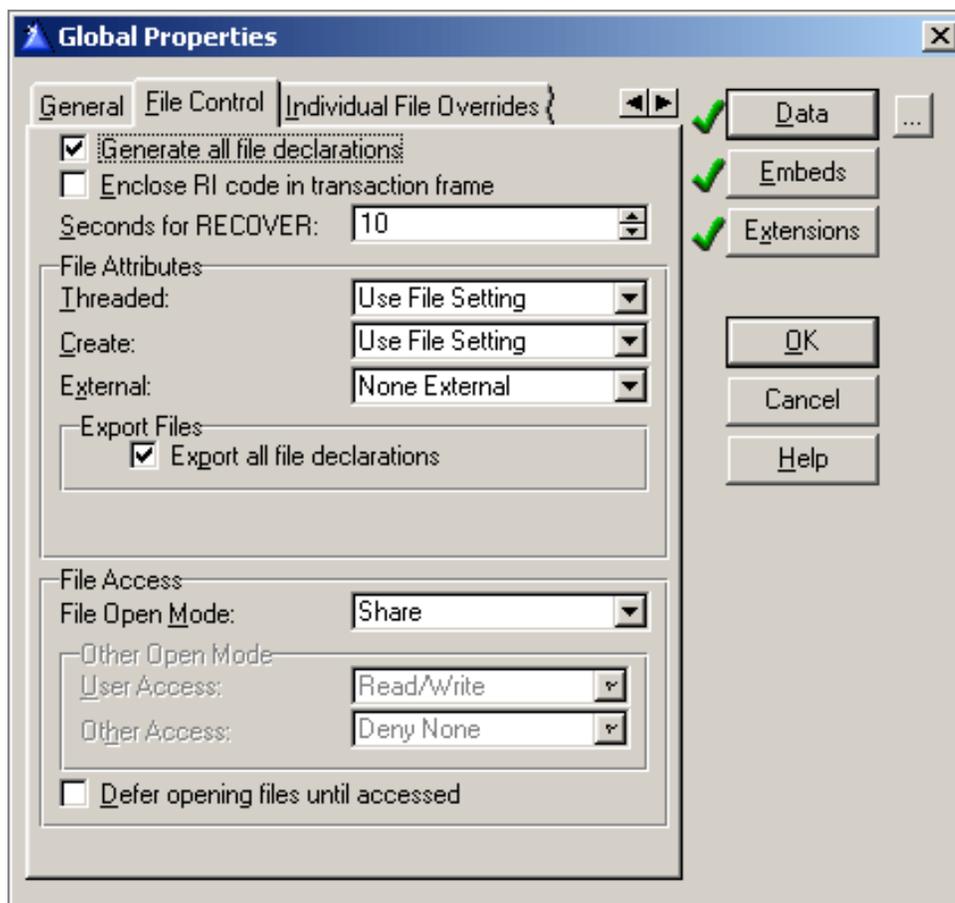


Figure 5. Setting Global Properties – File Control

Once you have done this, compile your application and you will have INV_SQL_DATA.DLL. This is just a basic approach to splitting up an application – further information is available in a number of other [Clarion Magazine articles](#).

Modifying your EXE

Now you need to connect INV_SQL_DATA.DLL to INV_SQL.EXE, and in doing so you must remove the data component in INV_SQL.APP. Close the INV_SQL_DATA.APP application and open INV_SQL.APP.

Step 1: Choose Application|Insert Module.

Step 2: Select a Module Type of ExternalDLL.

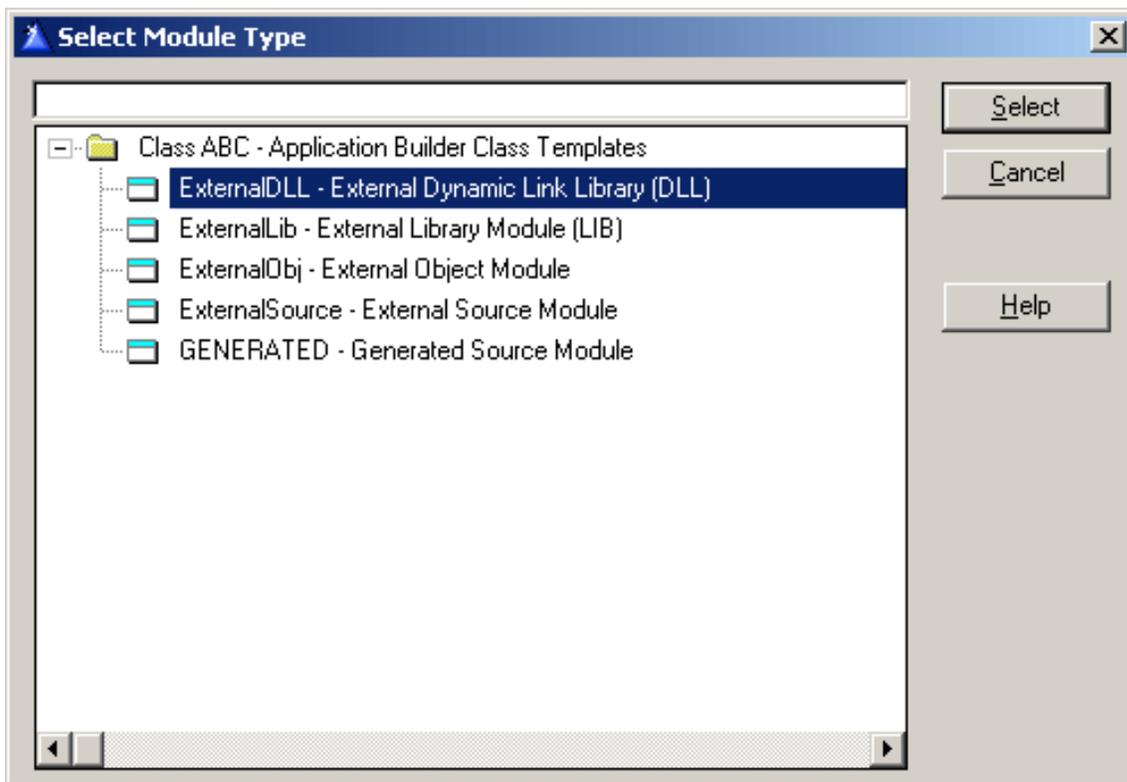


Figure 6. Selecting Module Type

Step 3: Name the module library, in this case INV_SQL_DATA.LIB.

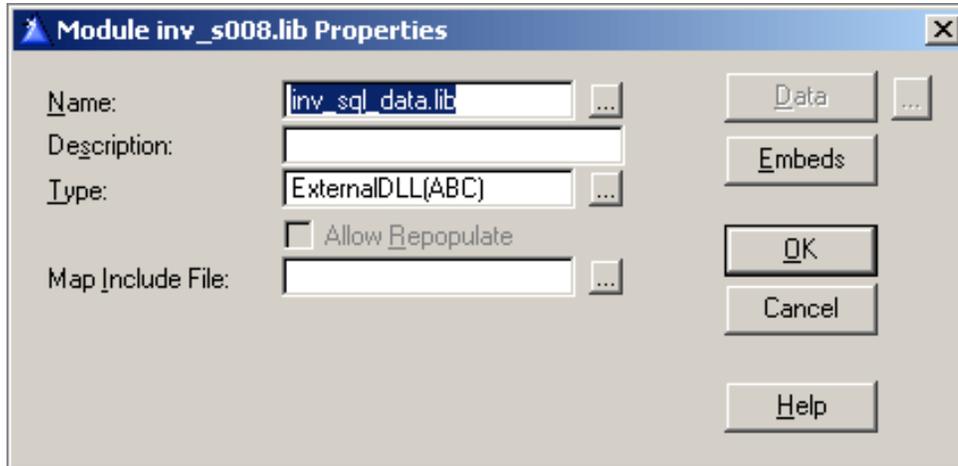


Figure 7. Module Name

Once you are done, you will have to modify the Global Properties.

Step 4: Global Properties - General. Check Generate template globals and ABC's as EXTERNAL. This doesn't remove the data declarations; instead, it declares them as EXTERNAL, which means they won't be allocated memory. The app must reference a LIB for a DLL that exports these declarations, which is what the data DLL does.

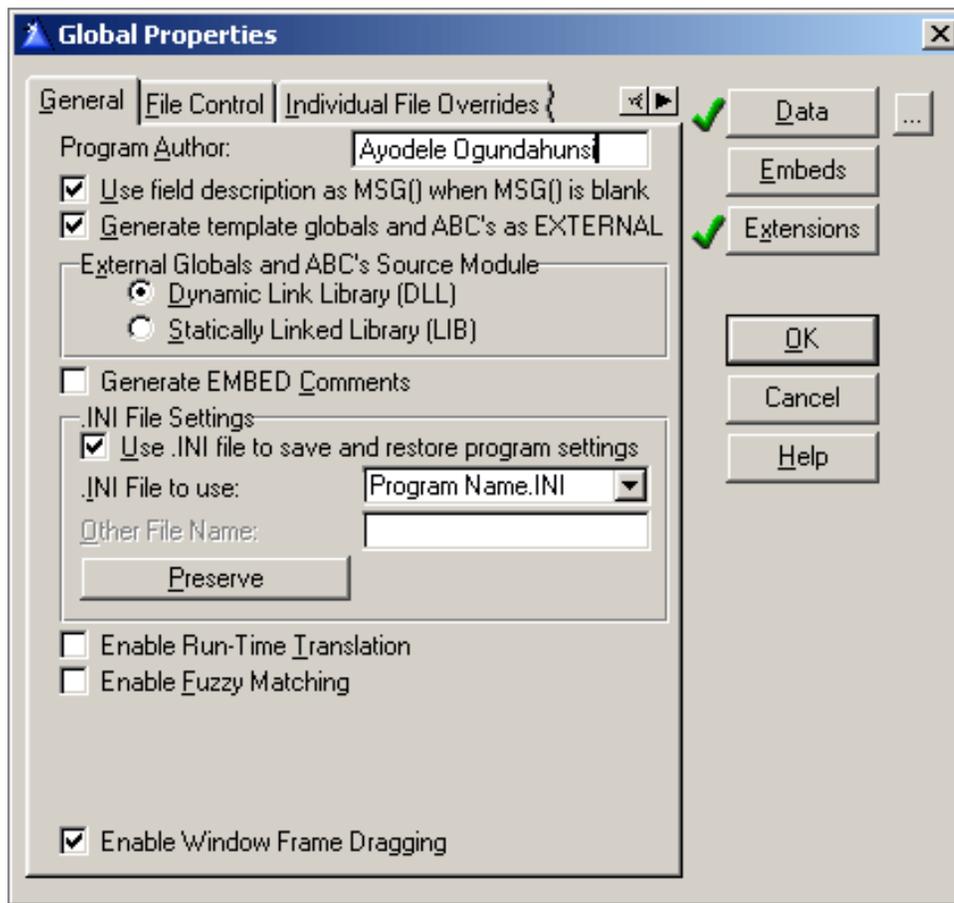


Figure 8. Global Properties – General (EXE)

Step 5: Global Properties – File Control. Uncheck Generate all file declarations, which means that this app will only declare the file declarations it actually uses. Also uncheck Enclose RI code in transaction frame - I always uncheck this since I enforce RI at the server

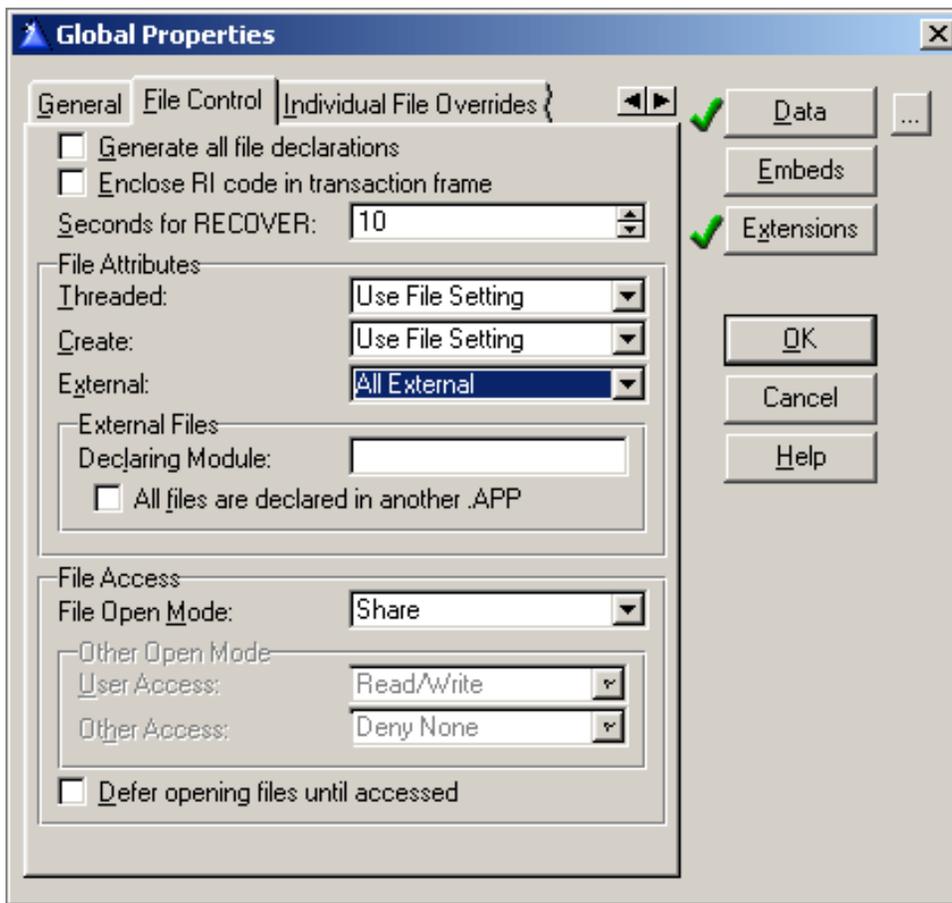


Figure 9. Global Properties – File Control (EXE)

While compiling, it is possible you will get an "Unresolved External MSSQL in inv_sql.obj" error, as shown in Figure 10.

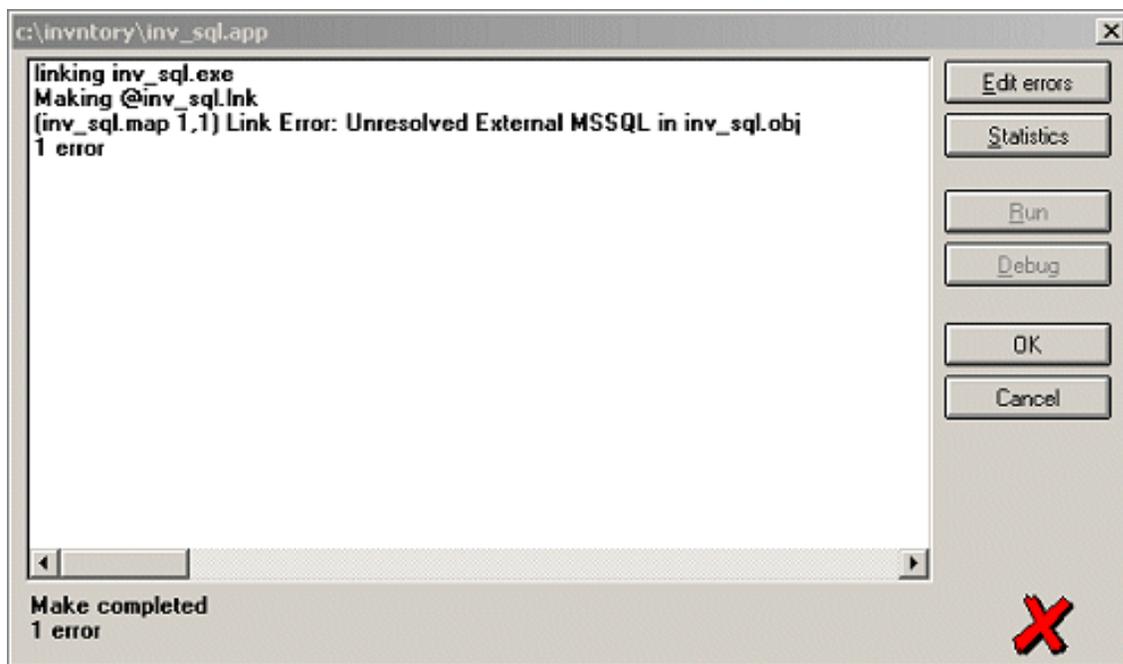


Figure 10. Unresolved external symbol compile error

It seems whenever you uncheck Generate all file declarations, the IDE removes the Database Driver Library for MSSQL Server. If this happens, just go to your Project Editor and add the driver, as shown in Figure 11.

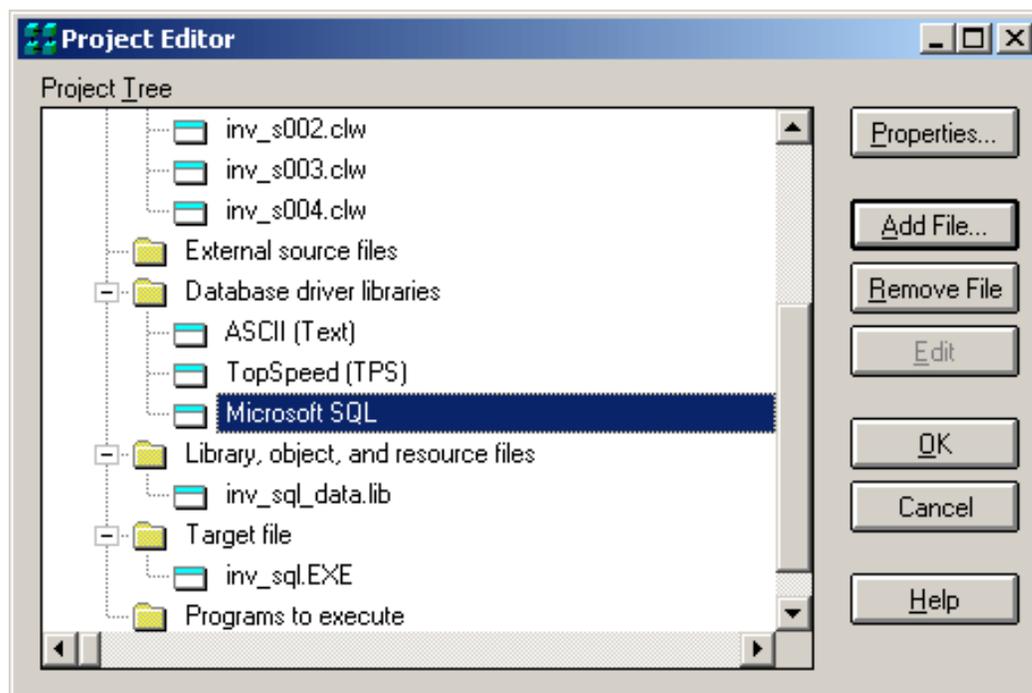


Figure 11. Adding MSSQL Driver

Now compile again - the error should go away.

In the [next article](#), I will show you how to use the supplied template to integrate the INV_SQL.EXE and INV_SQL_DATE.DLL applications, and make calling stored procedures very easy.

[Ayo Ogundahunsi](#) presently lives in Henderson, Nevada, about ten minutes from Las Vegas. He works for [Impac Medical Systems Inc.](#), the leading company in cancer therapy software (written in Clarion). Impac has its headquarters in Mountain View, California. Ayo is married to Ayodola, and they have two boys, Darren and Joshua.

Reader Comments

[Add a comment](#)

INVEST
in your own abilities

Clarion
magazine

[Ads](#) · [Comments](#) · [Writers!](#) · [Privacy](#) · [Contacts](#) · [PDFs](#) · [Freebies](#) · [Open Source](#)

[Topics](#) > [Design & Development](#) > [Conversions](#)

Converting The Inventory Example - Calling Stored Procedures (Part Two)

by Ayo Ogundahunsi

Published 2002-10-31

In my last article, [Converting the Inventory Example – Calling Stored Procedures Part 1](#), I showed how you can split the inventory application into an EXE and a DLL. Now it's time to begin implementing stored procedures. As part of this series, I have included a template to make integration to SQL Server a lot easier, and that's the subject of this article.

The AyoMSSQL Template

Before you can use the AyoMSSQL template you need to register AyoMSSQL.tpl (see the link at the end of this article). I will not go into full details about the inner working of this template, rather, I will describe how to use it.

The key features of this template are as follows:

1. It allows you to choose the method of data access from Clarion to SQL Server. (See Step 3).
2. It generates the connection string. You can define the variables without embedding any code.
3. It generates SQL code for creating a "Dummy" table.
4. It generates code to save your connection string settings in an INI file.

Implementing the SQL template – INV_SQL_DATA.APP

The first place to integrate the template is at the data layer (which I described in [Part 1](#)).

Step 1: Add the global extension to INV_SQL_DATA.APP.

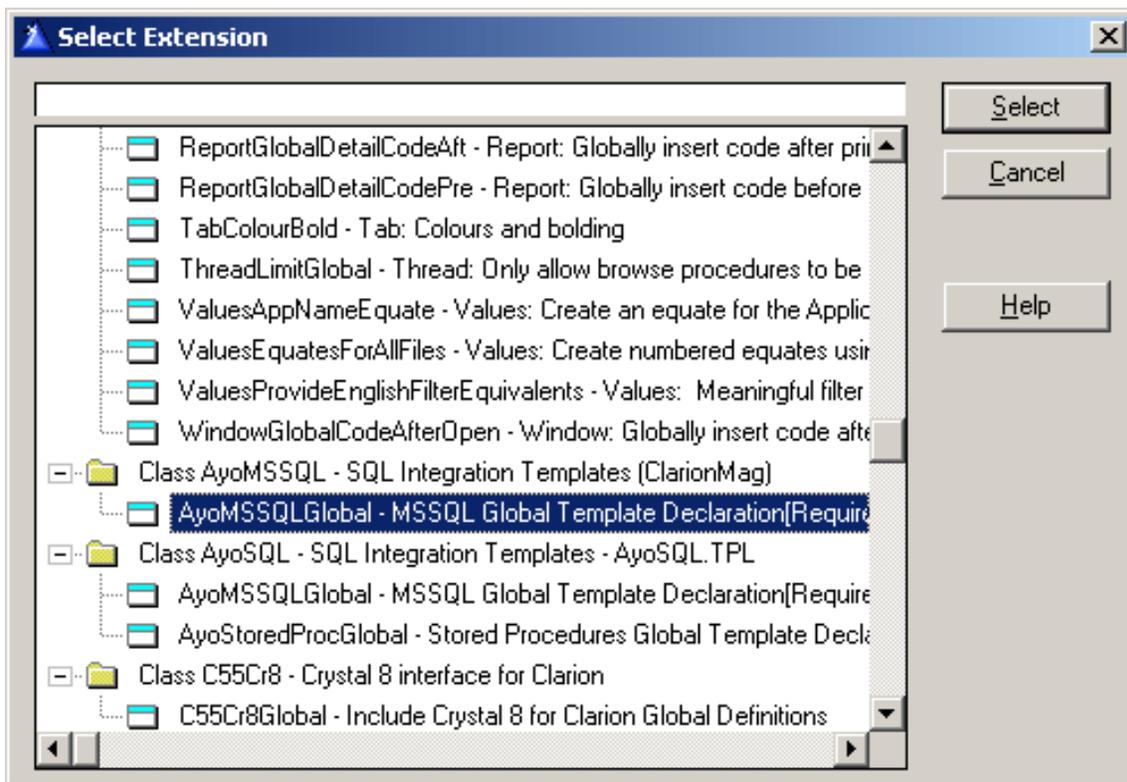


Figure 1. Global Extension

Step 2: Set the global properties. The first tab under the Global Extension template allows you to set Connection Properties, Connection Values, etc.

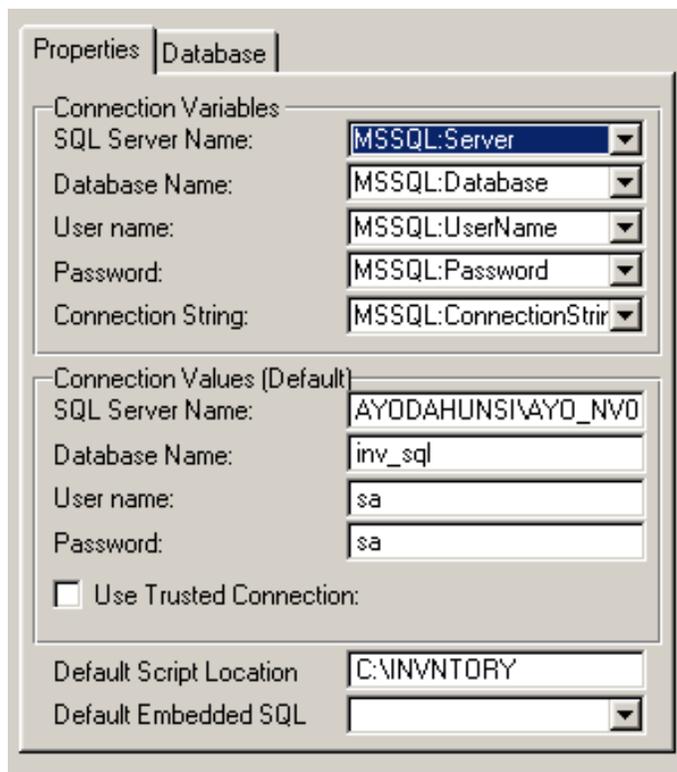


Figure 2. Global Extension - Properties

Connection Variables: These contain template created/generated variables that are used to form the connection string. Though you have to ability to change the name of the variables, it isn't necessary to do so. Note that all global variables, including those defined in the dictionary, are available in the drop list. However, since my connection string variable is now going to change from GLO:ConnectionString (as I defined it when I started this series) to MSSQL:ConnectionString, I need to reflect this change in the Owner Name field for Table properties of every MSSQL Table defined in the dictionary. Figure 3 shows the required change for the INVHIST table.

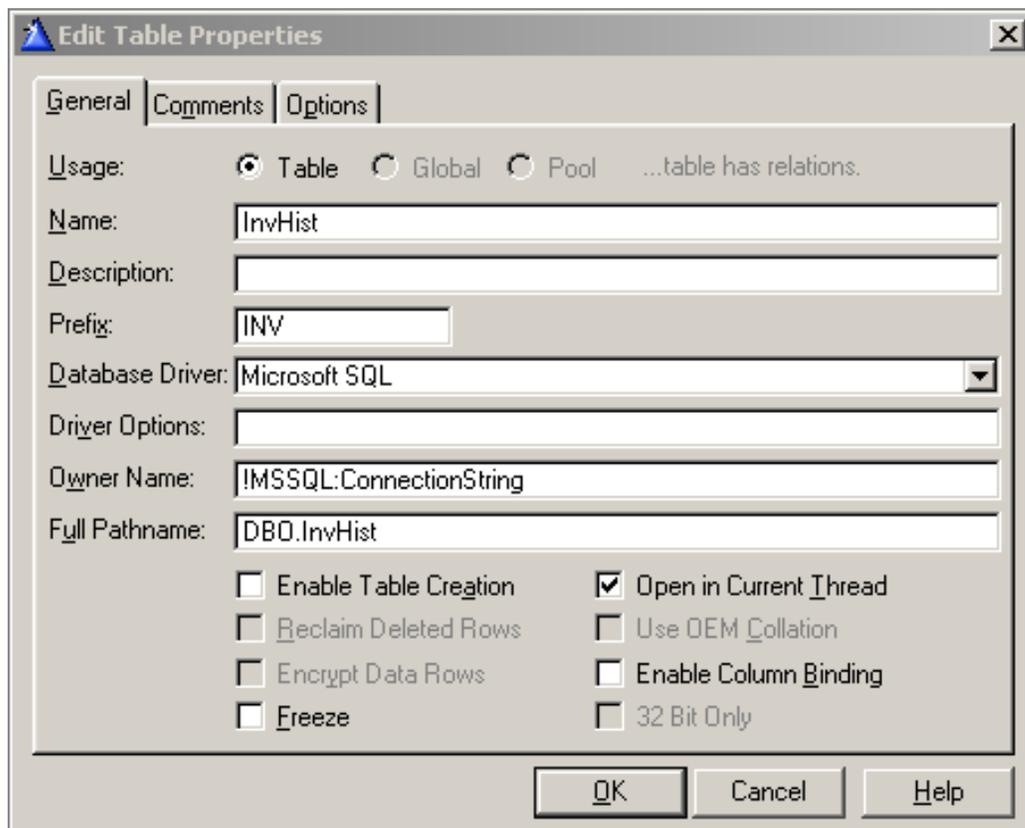


Figure 3. Table Properties

Connection Values: The values entered here (see Figure 2) will probably serve well during testing. However, for deployment, you would need to make these values updateable. That is why the connection variable described above becomes useful – you can write your own code to prime the variables.

Trusted Connection: As a recap, a trusted connection simply means that SQL Server will trust Windows NT/2000 to verify the user's password. Therefore on a Windows 98 machine this will not work. When Trusted Connection is checked, the Username and Password fields become disabled. This could be useful if you use Windows Authentication Mode in setting up your SQL Server (see my article [Migrating The Inventory Application To SQL Server - Part 1](#) for information on authentication modes).

Default Script Location: Some scripts will be generated in the course of using this template. They will be output in the location defined here.

Default Embedded SQL: The procedure selected here contains code that calls the `PROP : SQL` statement. `PROP : SQL` is never called anywhere else in this application. The main advantage is that you can isolate all your calls to SQL Server. There are two extension templates that generate this procedure, and which I'll explain a little later.

Step 3: Global Extension – Database. The second tab contains database related settings. These settings affect how data is returned (when stored procedures are called), and how tables are accessed during startup.

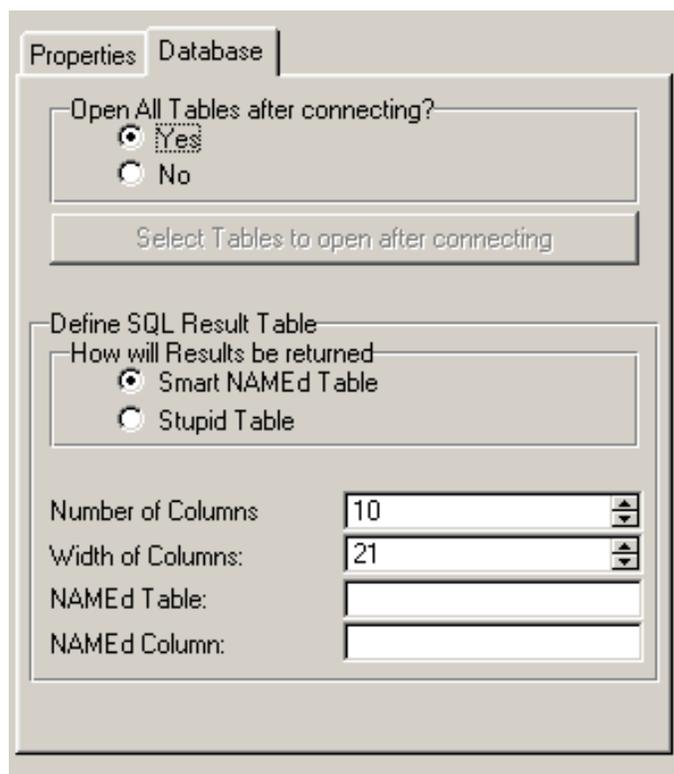


Figure 4. Global Extension - Database

Open all Tables after connecting: In Relational Database Management Systems (RDBMS), all tables are treated as one single unit – the database. What this means is that you don't really have `CLOSE`ing and `OPEN`ing of tables during the life of an application like you see in flat file systems. Clarion gives you the option of opening your tables when they are first accessed. If you decided to set up your system this way, a browse that displays information from a very big table (that is, many fields/columns) will take some time to load. This setting forces all the tables to be opened during startup. Psychologically, a user can bear with an application taking some time to load, rather than a browse loading very slowly when the application has been in use for some hours. This setting will display the table name in the title bar as every table is being opened. You can also decide not to force all tables to be opened during startup.

How will Results be returned: When returning results from SQL, you need an intermediate structure to catch the result. This can be a VIEW, or a TABLE. I will not get into details of what the code looks like, however, I will mention the logic behind this.

StupidTempTable Theory

Several years ago Troy Sorzano publicized what he called the StupidTempTable Theory (Editor's note: you can still read Troy's description via the [Wayback Machine](#)). This theory simply means you use a physical table to capture the results from a SQL backend. The StupidTemp Table approach in this article has been revised. Though my approach still requires a table on the server, I use a VIEW structure to display the results. Also, the column's data type is CSTRING. Clarion's automatic type conversion makes it easy to retrieve numeric data into a string data type and, the NAME attribute is not used in any column. In order to make this work, you will have to create a table (for the returned result set) on SQL Server. In my template, this VIEW structure is passed as a parameter to a common procedure that is created by a procedure template.

During compilation, a script file is generated and saved in the directory you specified under Default Script Location (Step 2 above). It is the "Stupid" or Dummy table structure located in SQL Server. You will have to run this script via the Enterprise Manager or OSQL.EXE before you run your application in order to create the table in SQL Server.

The generated script looks like this:

```
if exists (select * from dbo.sysobjects
where id = object_id(N'[dbo].[SQLRESULT]')
and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[SQLRESULT]
GO
CREATE TABLE [dbo].[SQLRESULT] (
  [Column1] [varchar] (21) NULL,
  [Column2] [varchar] (21) NULL,
  [Column3] [varchar] (21) NULL,
  [Column4] [varchar] (21) NULL,
  [Column5] [varchar] (21) NULL,
  [Column6] [varchar] (21) NULL,
  [Column7] [varchar] (21) NULL,
  [Column8] [varchar] (21) NULL,
  [Column9] [varchar] (21) NULL,
  [Column10] [varchar] (21) NULL,
  [ID_SQLRESULT] [int] IDENTITY (1, 1) NOT NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[SQLRESULT] WITH NOCHECK ADD
  CONSTRAINT [PK_RESULT] PRIMARY KEY CLUSTERED
  (
    [ID_SQLRESULT]
```

```
) ON [PRIMARY]  
GO
```

Smart NAMED Table Theory

The Smart NAMED Table Theory borrows some concepts from Troy's method in the use of the NAME attribute. However, no physical file has to exist in order for this approach to work. The basic concept is to fool the Clarion SQL driver into thinking that a table exists. The driver always compares table definitions in the dictionary with what exists on SQL Server, so if you give tables and fields external names that match names existing in your SQL Server, the driver will think the table actually exists. (See Clarion help for explanation on using the NAME attribute). In the template, a table (*FILE) structure is passed as a parameter to a common procedure created by a procedure template.

Number of Columns: (See Figure 4) This controls the number of columns being returned by your SQL query. If you want to execute a stored procedure that will return 15 rows of data, then you have to increase this to 15.

Width Of Columns: If you are going to be returning text that will be more than 21 characters in length, then you have to increase this size.

NAMED Table: This is the name of any existing table. Note that a table structure will be created that will be NAMED based on what you type here. However, the table will not be physically created on the SQL Server.

NAMED Column: This is any column that belongs to the table defined in the previous field.

Step 4: Extension Template – initialization code. The code that establishes a connection to SQL Server is automatically generated by this template. The variables used to create the connection string come from steps 2 and 3.

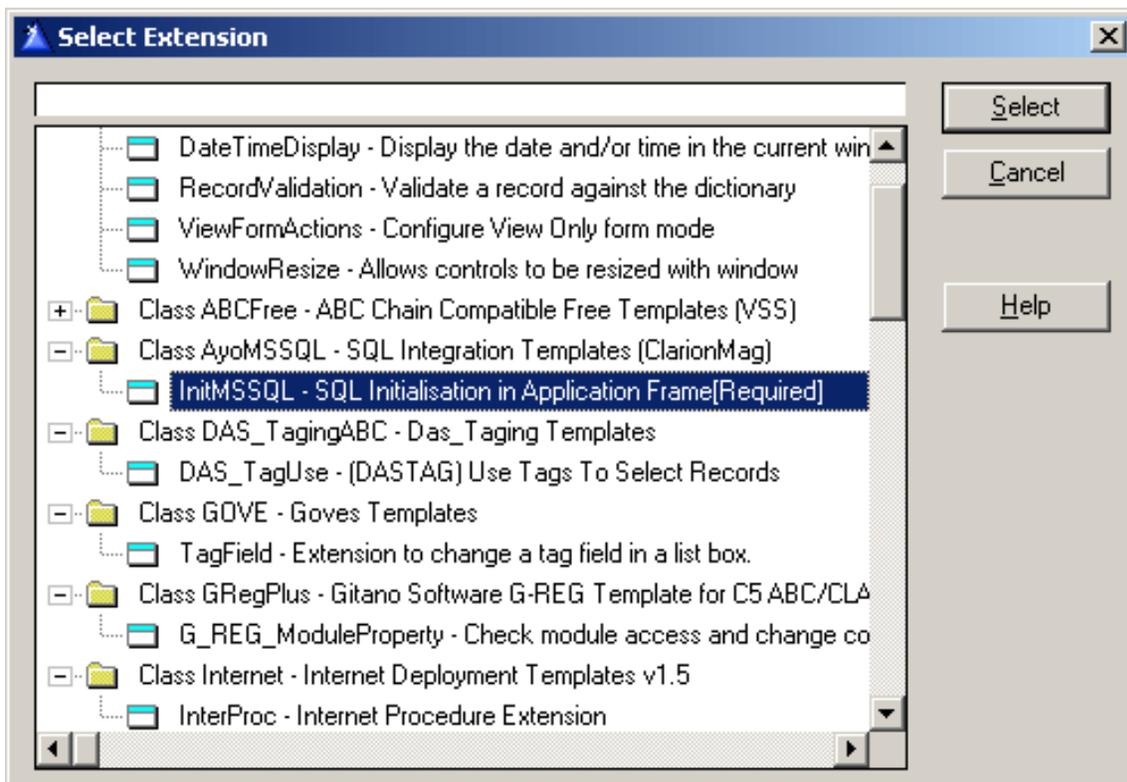


Figure 5. Extension Template – Initialization Code

The best approach in using this template is to create a source procedure, and then add this template as an extension to the source template. In this example, I will call this source procedure MSSQLConnector.

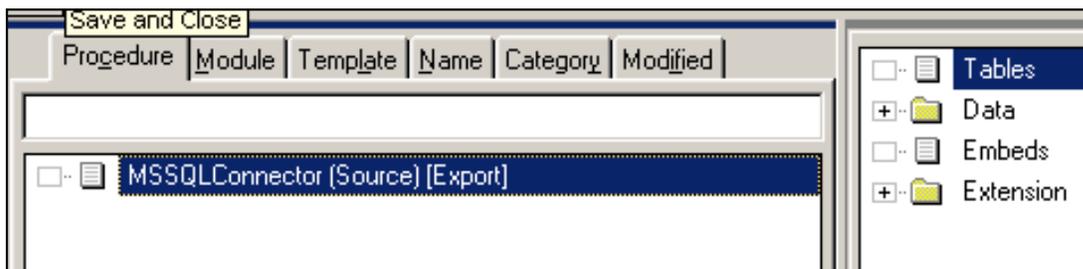


Figure 6. Extension Template - Connector

This procedure will be called in the Main frame of INV_SQL.APP.

Step 5: Procedure Template – ExecuteSQL. As I mentioned in Step 2, the code that calls PROP : SQL has been isolated to one single procedure. This procedure can be created from a procedure template supplied with this SQL integration template.

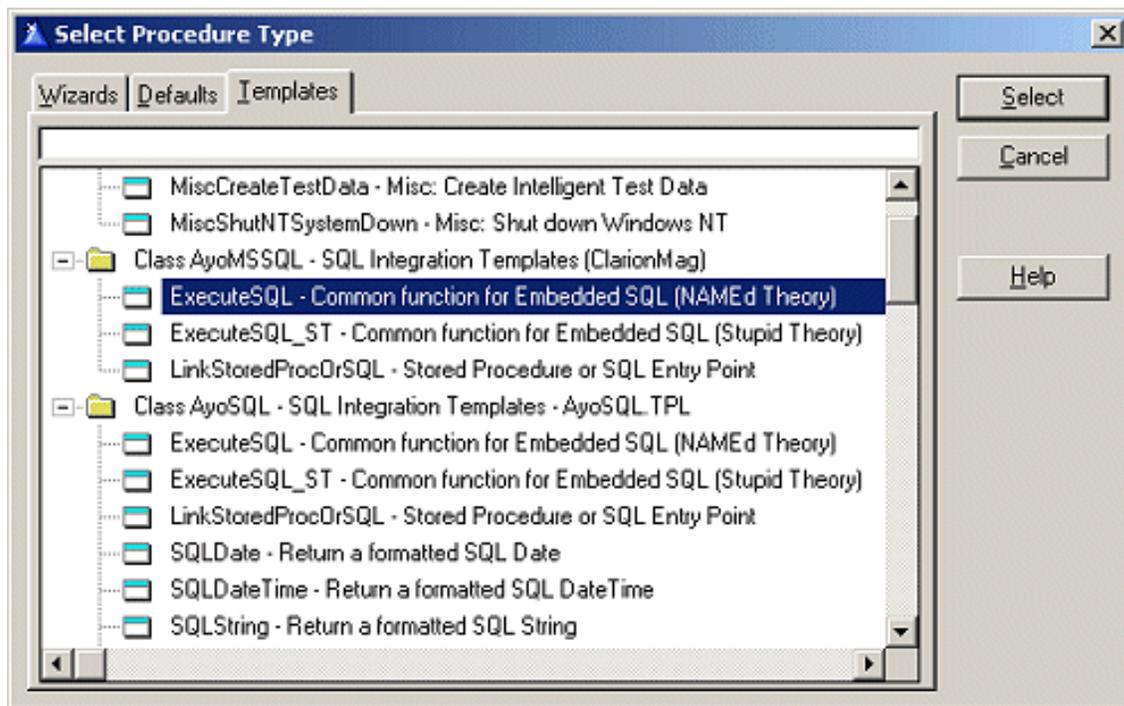


Figure 7. ExecuteSQL – PROP:SQL encoding

These two procedure templates basically do the same thing. The only difference is that one makes a call to the server using the Stupid Table Theory, while the other uses the NAMEd Table theory. I have named the procedures I created using these templates `SendsSQLString1` and `SendsSQLString2`

So far, everything that has do with a connection or any call to SQL Server has been restricted to `INV_SQL_DATA.APP`, which is the initial idea of grouping such functionality under the Data Access Layer.

I can now go ahead and compile `INV_SQL_DATA.APP`.

Implementing the SQL template – INV_SQL.APP

Step 1: Adding External Procedure(s). The first thing I need to do here is add the `MSSQLConnector` procedure as an External Procedure.

Step 2: Removing Tables from File Schematic. In [Part 3 of Migrating The Inventory Application](#), I added the tables `INVHIST`, `PRODUCTS`, `VENDORS`, and `ZIPCODE` to the table schematic of the main frame. These have to be removed since the external procedure `MSSQLConnector` handles this functionality.

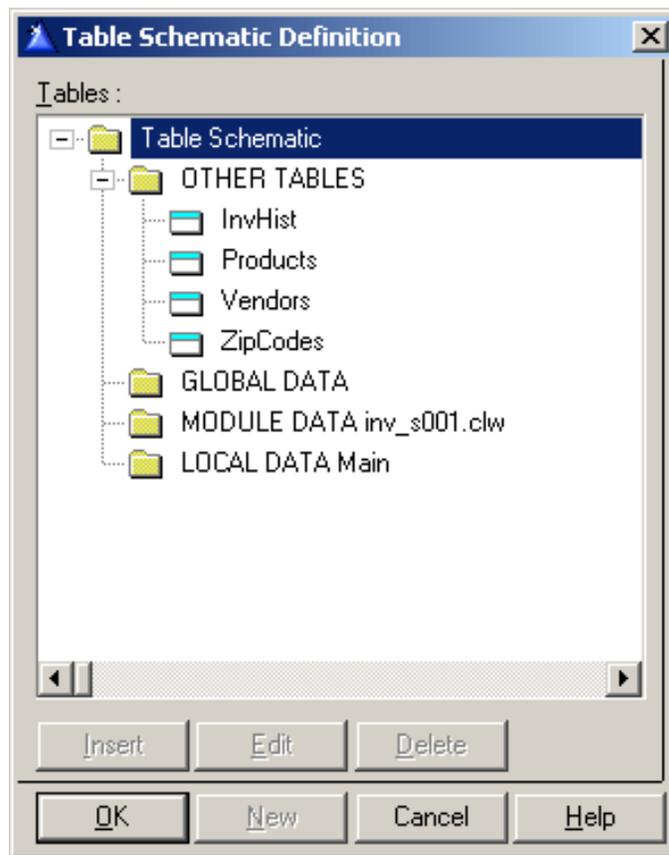


Figure 8. Removing Tables from File Schematic

Step 3: Adding the MSSQLConnector procedure. Comment out or delete the embedded code where a value is assigned to GLO:ConnectionString, and then add MSSQLConnector to an embed point immediately after Open the Window.

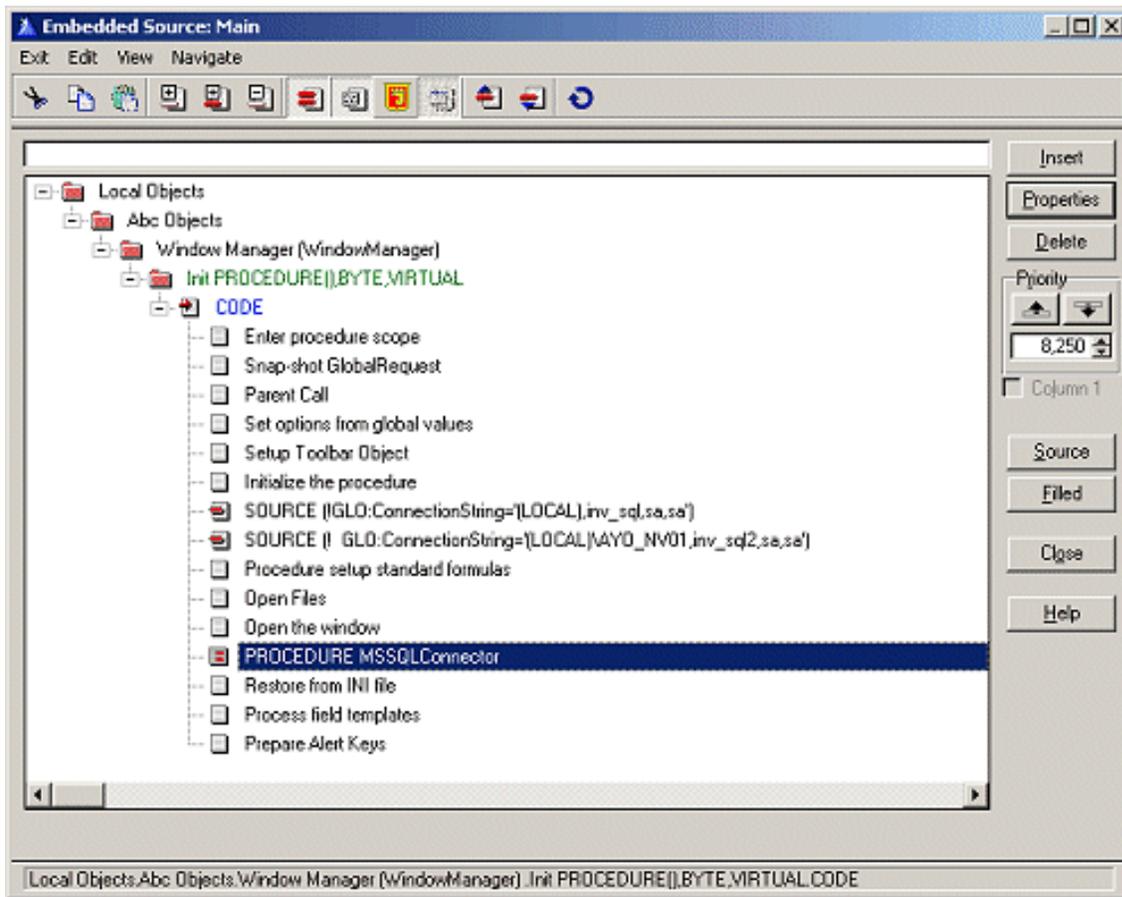


Figure 9. Adding MSSQLConnector to embed

In this article I have demonstrated how to use the supplied template. Next week I will show you how to call a stored procedure.

[Download the source](#)

Ayo Ogundahunsi presently lives in Henderson, Nevada, about ten minutes from Las Vegas. He works for [Impac Medical Systems Inc.](#), the leading company in cancer therapy software (written in Clarion). Impac has its headquarters in Mountain View, California. Ayo is married to Ayodola, and they have two boys, Darren and Joshua.

Reader Comments

[Add a comment](#)