Reborn Free

published by CoveComm Inc.

Search

Clarion MAGAZINE

October 2000 Index

If you're like most Clarion developers, you've at least considered the possibility of developing web applications. Since Clarion 2003, there's been out of the box support for web apps and it's not that difficult to hang something out on the Internet. Of course, life is hardly ever that simple. The problem is that Clarion developers, like other developers, face a bewildering array of web development options, some of which may not be immediately obvious. Part 2 of 2.
(Oct 24,2000)

Feature Interview: Mark Riffey
Mark Riffey is President and founder of Granite Bear Development. Granite Bear's Photo One studio management program has been named the HOT 1 award winner by the trade journal of the Professional Photographers of America. Mark spoke with Dave Harms, Clarion Magazine's editor and publisher.
(Oct 24,2000)

The Clarion Advisor: Copying Browse Boxes Between Procedures
Sometimes you need to make a copy of a browse box in another procedure. Unfortunately, Clarion's support for object-oriented development doesn't extend to deriving one browse box from another, but you can accomplish the task with a bit of cutting and pasting.
(Oct 31,2000)

Give It a Nudge: Adjusting Report Position at Runtime
As good as printers now are, there are inevitable differences between brands and models that make it difficult to consistently print within the bounds of the printable area. Lee White and Steve Parker have the answer: Give that report a nudge.
(Oct 31,2000)

October 2000 News
Clarion news, notes, and happenings from around the globe.
(Oct 31,2000)

Main Menu | Log In | Subscribe | Links | FAQ | Advertising

Reborn Free
CLARION online
published by CoveComm Inc.

Search

Main Page

COL Archive

Log In
Subscribe
Renewals

Frequently Asked Questions

Site Index
Article Index
Author Index
Links To Other Sites

Downloads
Open Source Project
Issues in PDF Format
Free Software

Advertising

Contact Us

Clarion MAGAZINE

An Introduction To Writing Templates

by John Morter

Part 1 of 2

One of the features of Clarion that sets it apart, and ahead, of competitive RAD products is its template technology. It was this concept that first attracted me to Clarion, back in the old Clarion 2.1 for DOS days when templates were called "models". Templates enable consistent, tested and "clever" code to be generated very quickly and without need for the developer to necessarily understand all of it (or any of it!).

The great advantage of templates, over the more typical wizard code-generation approach, is that they provide much more than just a one-shot starting point. By this I mean that the developer can generate an application (typically using the Application Wizard or Wizatron) and then go back and modify the characteristics, features and functions of the application just by changing attributes of templates, via their interfaces. And this can be repeated again and again, refining the template configuration each time until the required result is achieved, with minimal need to touch any "real" code (as is well exemplified by many of the sample applications that ship with Clarion).

Templates provide the driving force behind the application generation stage, churning out reliable code that complies with the developer’s high-level concept of the application design. They can also be used to accomplish menial, repetitive and verbose tasks, freeing up the developer to spend more time on the "fun bits". Plus, the developer can write her/his own templates to extend the functionality available to the Clarion IDE and to add personal "flavour" to generated applications.

This all sounds ideal, doesn’t it?! But, how to get started? How to make sense of all that weird *#command* syntax? The aim of this article is to answer these questions, and help you with writing your own templates.

> **Note**: This article refers specifically to the ABC templates, but it also applies generally to the legacy template set. The template language itself is independent of either approach.

Dipping Your Toe In

A good starting point is having an idea for a simple task you'd like to automate. You've probably all heard the exhortation, "*If you write the same bit of code more than once, put it in a template*." That may seem a little extreme, but if you have a bit of hand code that you find yourself manually inserting into every application, or (worse still) into many procedures, then you already have a good candidate for a template.

I've been paying a bit more attention lately to the various Clarion related newsgroups (well recommended by the way - what an amazing wealth of help and information). I came across a request for assistance in creating a template to implement some logic to be triggered when a button is pressed. I'll use this example to give you something to get your teeth into.

The approach I'll take is to cover technical theory and command syntax only as needed. There's absolutely no necessity to understand a great deal about the template language in order to be able to exploit it. You'll see that my own approach is to borrow, adapt and learn by example.

Template Types

First a bit of high-level theory: all templates are of one of four different types.

- You'll see a list of *Procedure* templates when you choose the starting point for a "ToDo" procedure in the Application Generator. A standard example is the *Window* procedure template.
- An *Extension* template adds new functionality to an application procedure, independent of any window control. It can generate code into one or more embed points. A standard example is the `DateTimeDisplay` extension template.
- A *Code* template is the simplest type. It usually generates code into the current embed point (i.e. the embed point into which you directly insert the template), but it can also embed code into other embed points. A standard example is the `LookupNonRelatedRecord` code template.
- A *Control* template is associated with a specific window control (or set of controls) - it usually both places a control on the window and generates code to support the control. A standard example is the `SaveButton` control template.

  Tip: I have paraphrased here from the *Template Language Overview* section in Clarion Help - see the *Template Types* topic for more detail.

The general aim of the example I'm using in this article is to execute some logic when a button is pressed. As you will see later, the specific intent is to put an "Apply" button on a Form that causes the pending update action to be applied.

A button is a window control, so the decision is a "no brainer" - a Control template will suit best.

Beg, Borrow Or Steal

My approach at this point is to ponder over where I've seen something like what I want to achieve - and a very good starting point is always with the standard shipping templates. So check out the standard control templates to see what you can find.

Start by getting yourself into the Window Formatter, then select the Control Template item from the Populate menu - this lists all the Control templates in registered template sets, including any templates you may have acquired from third parties (or wrote yourself).

- `FieldLookupButton` is worth a look - it puts a button on the window
- so does `BrowseSelectButton`

So, where does onefind these templates? Easy, they're all located in the *Template* directory (surprise, surprise!). Take a look - you'll find both *<Template>.TPL* and *<Template>.TPW* files.

> **Tip**: To reduce a lot of the clutter in the *Template* directory, I shift all graphics files into the *Images* directory (where they're still included in the standard redirection search path). This makes it a bit easier to find your way around the templates.

<Template>.TPL files are "template chains" - they provide a "header" for the collection and management of a template set, (and they may also contain set-up template code). For example, see *ABChain.TPL* - the template chain header for the ABC templates. Towards the bottom of this file (around line 570) you'll find #INCLUDE statements referring to *<Template>.TPW* files.

The *<Template>.TPW* files contain the actual templates themselves. (Note: I'm generalising here; TPL files can contain actual templates too, but that's not the defacto Clarion standard.)

By convention, the files making up the ABC template set are all prefixed by "AB". Furthermore, all ABC wizards (which are just templates) are prefixed with "ABW". Individual .TPW files are just containers for related templates.

Search all *AB*.TPW* files, looking for `FieldLookupButton` - You'll find it in *ABContrl.TPW*.

> **Tip**: I highly recommend getting yourself a good text-searching tool for this sort of work - one that will quickly scan through a bunch of selected text files for a specified string, and then present the results to you in context. I cringe every time I see someone trying to do this with Explorer's *Find* tool!
> (Editor's note: Carl Barnes' Clarion Source Search is one such utility.)

The main items of interest here are;

- the `#CONTROL` statement, which identifies all following code as belonging to the named Control template
- the `CONTROLS ... END` structure, which declares and describes the attributes of the window control (in this case, a `BUTTON` control)
- the `#PROMPT` statement, which gets input from the developer (That's you!)
- the `#ATSTART ... #ENDAT` structure, which looks "interesting" - I'll come back to this later
- the `#AT(...) ... #ENDAT` structure, which puts code into an embed point
- the rest is just "noise" at this stage - you can ignore it

> **Note**: Template language statements are prefixed by #, to distinguish them from standard Clarion language statements.

It looks like you have all the fundamentals you'll need here, but let's see what else can be found in the `BrowseSelectButton` template.

Search all AB*.TPW files, looking for `BrowseSelectButton` - You'll find it in ABBrowse.tpw.

The main items of interest here are;

- the `#CONTROL` statement, which identifies all following code as belonging to the named Control template
- the `CONTROLS ... END` structure, which declares and describes the attributes of the window control
- the `#PROMPT` statement, which gets input from the developer
- the `#ATSTART ... #ENDAT` structure, which, again, looks "interesting"
- the `#AT(...) ... #ENDAT` structure, which puts code into an embed point

There’s not much difference (from a high-level perspective) between these two templates. This tells us a great deal. There's a consistent approach being used - so it makes good sense to copy it!

Taking The Plunge

> **Note**: Language statements too long to fit on one line are split at the point where the ↵ symbol is shown.

First off, adapt the `#CONTROL` statement to be specific to the intent of your template. Say, something like:

```
#CONTROL(TriggerCodeButton,'Trigger some code when button ↵
is pressed'),DESCRIPTION('Code trigger button'),WINDOW
```

In this case, `TriggerCodeButton` will be the name of the new template you’re creating, analogous to `BrowseSelectButton` or `FieldLookupButton`.

The text immediately following the template name is appended to it, and the result is displayed when the developer steps through the Populate, Control Template, Select Control Template process. There’s a subtle difference between this text and the `DESCRIPTION()` text. The latter is displayed when viewing the list of extension and control templates via the Procedure Properties’ Extensions button, or viewing a list of code templates via the Procedure Properties’ Embeds button.

> **Tip**: You’ll notice the `BrowseSelectButton` declares: `DESCRIPTION('Select a Record from Browse on ' & %Primary)`."`%Primary`" is an example of a template symbol that will be expanded to add more specific meaning to the description. More on template symbols a bit later.

The `WINDOW` attribute makes the `TriggerCodeButton` `#CONTROL` template available in the Window Formatter, i.e. when the developer steps through the Populate, Control Template, Select Control Template process. (`WINDOW` is the default. The alternative option, `REPORT`, makes the `#CONTROL` template available in the Report Formatter.)

The `MULTI` attribute is very useful - it determines whether or not the `#CONTROL` can be selected multiple times for the same window. For example, notice that the `MULTI` option is declared for the `FieldLookupButton` (to allow multiple lookup buttons on the same window), but not for the `BrowseSelectButton` (for obvious reasons).

> **Tip**: For details of other options and attributes on the `#CONTROL` statement, see the Programmer's Guide (PG).

Using my borrow-and-adapt methodology, a good starting point for the requested template has been established. In Part 2 I'll show you how to continue with this *adaptation* approach to build upon the framework I've *borrowed*.

Download the source

John Morter is a member of the Victorian Clarion Users Group (Melbourne, Australia). His moneymaking day job doesn't actually involve Clarion (at least not officially), but Clarion occupies a lot of his spare time as a hobby to keep his techo-developer background up to date. He sails in the bay during the summer on his racing catamaran named Flat Chat, which is Australian slang for "at top speed" - or "at high velocity".

Reborn Free CLARION online

published by CoveComm Inc.

Clarion MAGAZINE

From The Publisher

The State Of The Mag

It appears that I've unintentionally alarmed some readers by taking a two week break at the end of September. Several of you have emailed me wondering why the disruption in the publishing schedule, and asking if something is wrong.

Hardly. As some of you know, Bonny and I celebrated our wedding last Christmas. We'd intended to go on a honeymoon shortly thereafter, but work obligations, the purchase of a house, and even the ETC conference intervened. And summers here in Winnipeg are usually so good that it hardly made sense to go far away then. So we booked two weeks in Spain and Portugal for the end of September, and yes, we had a most excellent time.

One thing I've discovered about publishing a magazine is that it's even more difficult to take time off than when working on a consulting project. I contemplated simply suspending publication for several weeks, and extending everyone's subscription, but I decided it would be better to continue with the normal level of content and publish a double issue in the middle of September. I also expected I'd need a little jetlag recovery time on returning (I was right), and as Clarion Magazine publishes four times per month, and October has five Tuesdays, I opted to resume publication on the 10th rather than the 3rd.

I realize that's meant three Tuesdays have passed without a fresh issue of Clarion Magazine. If the magazine were my sole occupation, I could probably have reduced that to two, but I'm also at present under contract to deliver another Java book, and I still do some Clarion consulting. The upshot is that although the schedule went a bit wobbly for a few weeks, the level of content has stayed within the usual parameters.

Finally, a word about the Clarion Online archives. I haven't forgotten about them, and they are still on the way. When we obtained the COL rights the only available archive was in PDF format, as the originals had been lost. For various reasons, including personalization of all the pages, the PDF cannot simply be posted as is, and conversion of the material is time consuming.

As well, the content in Clarion magazine has reached the stage where static pages take excessive effort to maintain, and the COL articles add to this problem. The next generation of Clarion Magazine is now under development, and the first phase will be a COL section with dynamically created pages, which will expedite the delivery of that material. In the meantime, if you have a specific COL article you'd like to see, email me at dharms@clarionmag.com and I'll post it as soon as possible.

Thank you for your patience during this temporary schedule upheaval. Normal service has resumed.

Dave Harms
Publisher

Reborn Free CLARION online
published by CoveComm Inc.

Search

Template Writing Made Easier: The Template Wizatron

by Steve Parker

There is a little-known Clarion utility, the Template Wizatron, which addresses a need many of us have. The need for it stems from the complexity of the template system.

When I first started using Clarion, the Model files were something of a mystery, not only to me but to many others. Repeat structures and Point fields (which made file browses, then known as Tables work), the fact that the Models were based on tokens and those tokens didn't always behave consistently, these among other things were real eye-crossers. Mike Hanson characterized these tokens as "magic tokens" because they magically did different things under different circumstances.

With the introduction of Clarion for DOS 3.0 much of the mystery was removed. Templates replaced Models and tokens disappeared, replaced by single purpose template symbols (variables). The entire code generation process was opened to view by developers. It became, if not quite easy, much easier to redefine standard application behaviors. A developer could modify the templates or create new templates which either extended or modified the behavior of the standard templates. New templates could be linked to the standard template chain or a whole new chain could be created.

For example, the standard Report template included a progress screen that showed page and line number. But if a report didn't find the first qualifying record until well into the processing of the file, this display did not change. With very large files, users could believe that the PC was hung. A progress bar, like that provided in the Batch template, seemed called for. Adding one (and debugging it) took an hour or two.

Editor's Note

SoftVelocity has made several minor changes to the Template Wizatron since this article was written (and, incidentally, previewed by SV staff). It is now called the Template Writer (a good move), and there have been some cosmetic improvements, including updates to reflect the new company name. All #PROMPT types are now supported. As well, the new version only supports Clarion 5.5.

Because the templates were essentially a set of conditional switches and straightforward Clarion code, they were easy to comprehend. With minimal study, it was not difficult to modify or create templates.

This all changed radically with Clarion for Windows. By Clarion for Windows 2.0, and even more so with the coming of the ABC templates, Clarion templates became virtually unreadable by mere mortals and many developers. (Arguably this got better with ABC, since much of the complexity was moved to the classes. But that assumes you could understand the classes. Unfortunately, this assumption is not often justified.)

Instead of seeing template files which appear to correspond to identifiable procedures, we see files which seem to correspond to controls. While, on reflection, this makes sense since a Windows procedure really is just a collection of controls, it is disconcerting to long-time Clarion developers. The template files themselves contain no recognizable Clarion code to speak of. While there are substitution symbols for functionalities I recognize like %LocatorType, %RecordFilter and %RangeField, there are also statements like NAME, FROM, DEFAULT and INSTANCE which, frankly, make no sense to me.

Examining the code for an individual control is a bit better. At least there, I can see things like #BOXED, #ENABLE, #SHEET that correspond to what I see on the template worksheets. But I also see statements like #PREPARE, #CALL, #DECLARE and #SET and somewhere between little and nothing that looks like code to make the control do something.

No wonder I gave up writing my own templates. And mind you, I wrote a complete set for CDD 3102 that implemented not only the report progress bar mentioned above but full parameter passing, parameter pass-through, batch start without confirm, browse-less recursive adds, queue based tagging, automatic display of copyright notices and a number of other features.

Fortunately, we have a number of third party vendors providing a wide variety of add-on templates. If I want tagging, transaction-like parent/child updates, security, prompted report redirection, automated file upgrading or any of a number of other commonly required features, chances are there is a commercial template available.

But what if I use the same bit of personalized code over and over and there is no third party template? Let's say that in *every* report I create I do not want the standard "no records" message. I *can* modify the standard templates or I would if I could find the embed in the template registry. In fact, there is no such embed in the Clarion template chain and, in ABC, this embed is actually in a virtual method. There is no ABC template embed and so I would have to derive the ReportManager or Error class.

Now suppose I want this functionality in only 75% of my reports ….

A template, implemented in a few mouse clicks, is preferable, preferable by far.

A Case In Point

I recently decided to stop deciding for my users whether or not to preview reports. Up until that epiphany, I had decided that preview made no sense for certain reports and, so, I did not offer it. All other reports were automatically previewed. Well, this really isn't my style. I think users ought to have a modicum of choice (just enough to keep them at bay and no more than that).

So, I implemented a FAQ posted by Russ Eggen about preventing print preview in code:

This is actually pretty straightforward in ABC.

In the ThisWindow.AskPreview in the first available embed point, simply add this little gem:

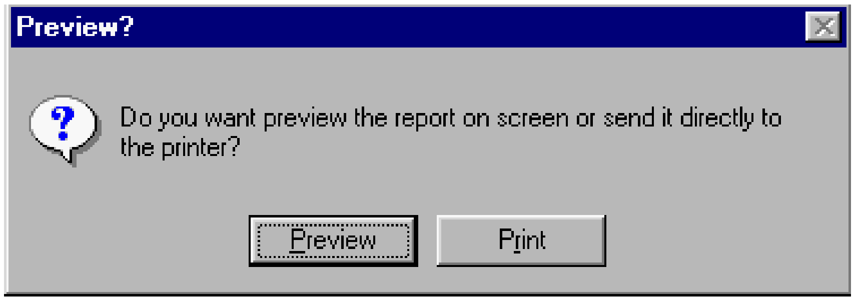
```
If UserNotWantPreview THEN SELF.SkipPreview = True.
```

The variable to control this would have to be defined by you and user accessible.

I had to make a few decisions about how to implement the FAQ.

First, I decided to use the `Message()` function to ask the user, at runtime, whether or not to preview or print (Carl Barnes' CW Assistant makes this easy):

Figure 1. Message() to user.



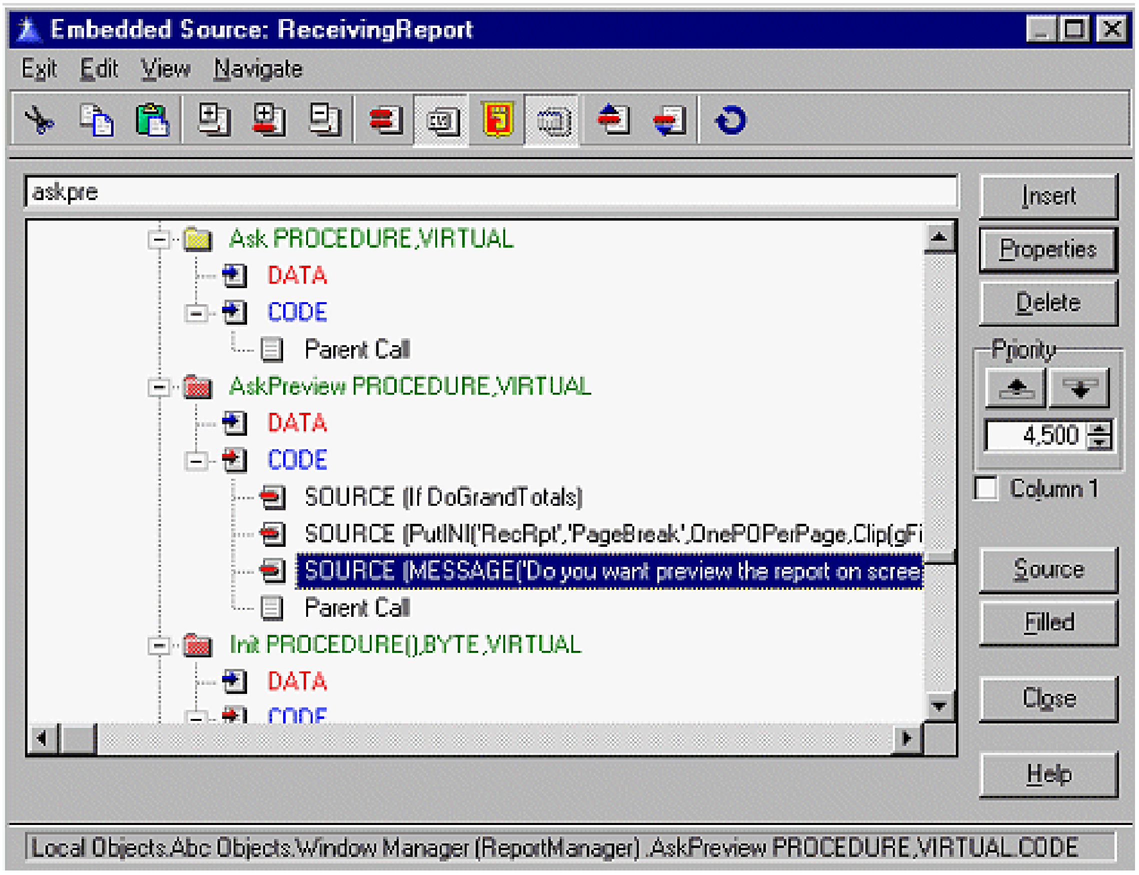
Do you want preview the report on screen or send it directly to the printer?

By using `Message()` I didn't need a procedure to return the user's choice. And I avoided configuration files – users always seem to forget where the configuration screen is hidden.

Second, I decided to prompt the user later than suggested, just before the Parent Call. Since `ThisWindow.AskPreview` is also where a grand total band would be printed, it seemed a bit more logical to print the totals and then ask the user.

Why ask here and not at the beginning of the procedure? Well, I think it odd to ask whether or not to preview a report and then have no records found. Especially on small files or fast machines, the time between the preview request and the no records notice can be negligible. Asking in `ThisWindow.AskPreview`, I *know* that there is a report to present. As the saying goes, salt to taste.

Figure 2. Placement of the code to ask.



```
askpre
Ask PROCEDURE,VIRTUAL
  DATA
  CODE
    Parent Call
AskPreview PROCEDURE,VIRTUAL
  DATA
  CODE
    SOURCE (If DoGrandTotals)
    SOURCE (PutINI('RecRpt','PageBreak',OnePOPerPage,Clip(gFi
    SOURCE (MESSAGE('Do you want preview the report on scree
    Parent Call
Init PROCEDURE(),BYTE,VIRTUAL
  DATA
  CODE
4,500
Local Objects.Abc Objects.Window Manager (ReportManager) .AskPreview PROCEDURE,VIRTUAL.CODE
```

(In the procedure pictured, I am also saving the user's preferences. But, that's another story.)

Design Considerations

Russ' FAQ assumes that a variable is primed earlier in the procedure and tested in `AskPreview`. You may use an INI file, a configuration file, a message function or another procedure but the variable needs to be primed before `AskPreview`, where it is tested.

In my implementation, the `AskPreview` method is going to be called even if there are no records to display. I need to trigger the `Message()` only if there *are* records. That means I need some indication of that state. I usually do this by priming a variable in `TakeNoRecords` (since this is called only if there are none) and testing it where needed (`AskPreview`, in this case). I also use this technique to trigger printing of a "No Records" band when printing end-of-day (i.e., unattended) reports.

These are just two variations on this theme. Either technique requires a local datum and an embed to prime it. Either is easily accommodated.

Yes, I know I should be using RPM or CPCS for this. But if I did, how would I overcome my fear of templates? How else would I have discovered …

The Template Wizatron

Introduced, quietly, with 5.0B,

> The Template Wizatron is a tool to help you add functionality to existing style files. Its purpose is to convert existing embedded source code extracted from an application and process them into template syntax.

I do not want to create a Wizatron Acorn or style file, which the Template Wizatron appears to be designed to do. But I do want to extract my embedded code and format it for use as a template. A template (or a file in "template syntax") should be a template (or easily changed to a template), regardless of the intended purpose.

In fact, the documentation later states that "The Template Wizatron creates a template." Ok. This is definitely worth a look-see.

Using The Template Wizatron

Here are the steps, as culled from the documentation, to create CODE or EXTENSION template "from a completed procedure":

1. Selectively export a procedure containing the desire features to a TXA.
2. Run the Template Wizatron (TW.EXE): Start|Programs|Clarion 5..5|Tools|Template Wizatron.
3. From the File menu, import the TXA.
4. From the Browse menu, browse and edit DATA, EMBEDS, CONTROLS, INFO and PROMPTS.
5. Create the TPW from File|Export (Write Template). You will get a preview of your template at this point.

Can it be this easy to get a template from my code?

You Pays Your Money, You Takes Your Choice

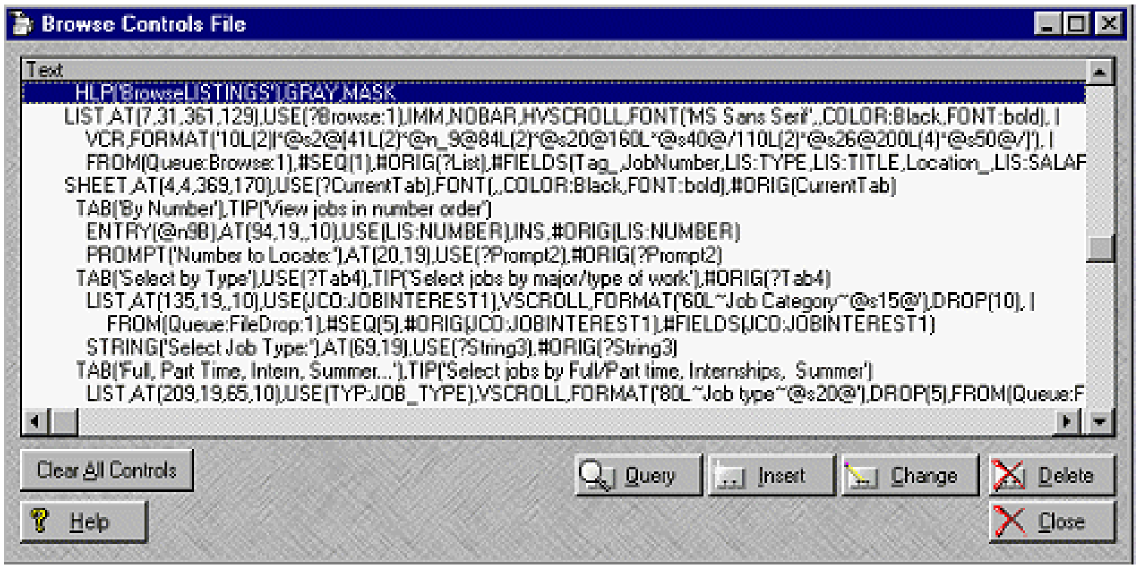
Following the instructions explicitly means that every embed and control in the original procedure ends up in the TXA. Of course, standard Clarion controls and data do not need to be in the template (but there's not a lot for this either). Neither do most of the controls populated from the dictionary have to be included (does the template really need to know what my aging report looks like to generate a "Do you want to preview?" template?). Similarly, embedded code and local variables that are not to be part of the template really needn't be in the TXA either.

Any controls and code which are in the TXA but which are not to be part of the final template have to be "edited" (step 4). That is, they have to be deleted.

The problem is that the editing capabilities of the Template Wizatron, I discovered, are not as flexible as those I am used to in the rest of the Clarion environment.

For instance, text cannot be block selected for deletion. Only one line at a time can be removed. And keep in mind that code for a single control can spread across more than one line. Each line needs to be deleted individually.

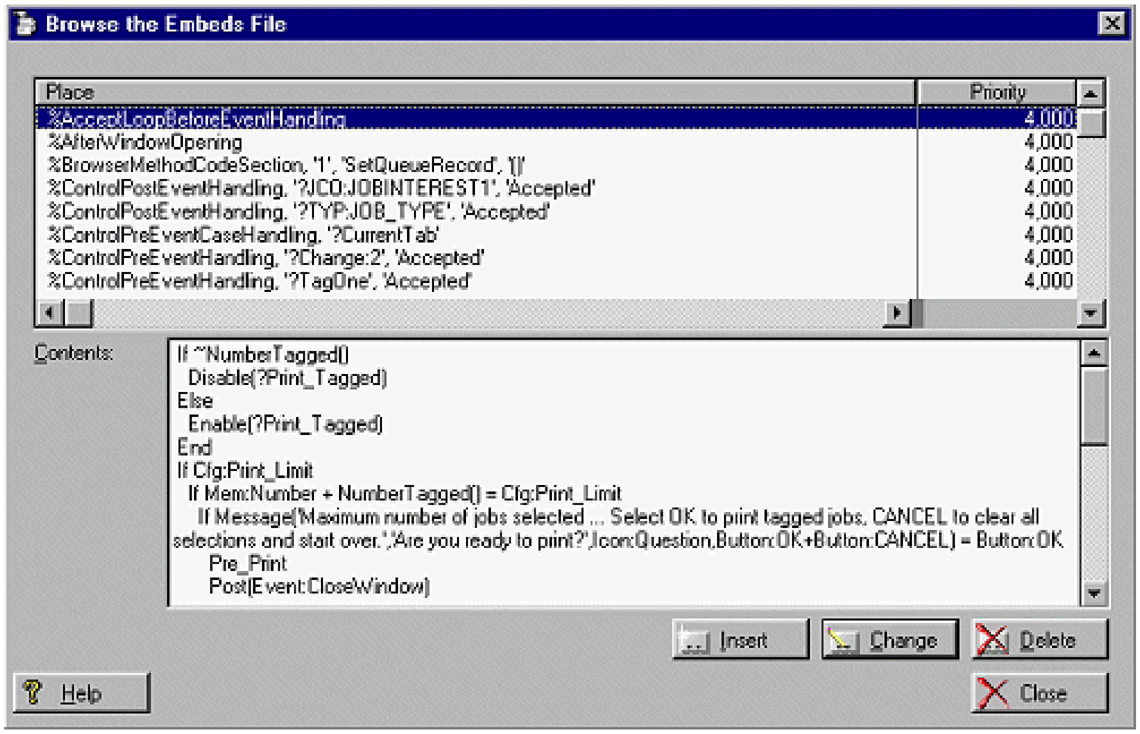
Figure 3. Controls list from TXA.



```
    HLP('BrowseLISTINGS'),GRAY,MASK
  LIST,AT(7,31,361,129),USE(?Browse:1),IMM,NOBAR,HVSCROLL,FONT('MS Sans Serif',,COLOR:Black,FONT:bold), |
    VCR,FORMAT('10L(2)|*@s2@[41L(2)*@n_9@84L(2)*@s20@160L*@s40@/110L(2)*@s26@200L(4)*@s50@/]'), |
    FROM(Queue:Browse:1),#SEQ(1),#ORIG(?List),#FIELDS(Tag_,JobNumber,LIS:TYPE,LIS:TITLE,Location_,LIS:SALAF
  SHEET,AT(4,4,369,170),USE(?CurrentTab),FONT(,,COLOR:Black,FONT:bold),#ORIG(CurrentTab)
   TAB('By Number'),TIP('View jobs in number order')
    ENTRY(@n9B),AT(94,19,,10),USE(LIS:NUMBER),INS,#ORIG(LIS:NUMBER)
    PROMPT('Number to Locate:'),AT(20,19),USE(?Prompt2),#ORIG(?Prompt2)
   TAB('Select by Type'),USE(?Tab4),TIP('Select jobs by major/type of work'),#ORIG(?Tab4)
    LIST,AT(135,19,,10),USE(JCO:JOBINTEREST1),VSCROLL,FORMAT('60L~Job Category~@s15@'),DROP(10), |
      FROM(Queue:FileDrop:1),#SEQ(5),#ORIG(JCO:JOBINTEREST1),#FIELDS(JCO:JOBINTEREST1)
    STRING('Select Job Type:'),AT(69,19),USE(?String3),#ORIG(?String3)
   TAB('Full, Part Time, Intern, Summer...'),TIP('Select jobs by Full/Part time, Internships,  Summer')
    LIST,AT(209,19,65,10),USE(TYP:JOB_TYPE),VSCROLL,FORMAT('80L~Job type~@s20@'),DROP(5),FROM(Queue:F
```

The same is true for embedded code:

Figure 4. Embed list from TXA.



Place	Priority
%AcceptLoopBeforeEventHandling	4,000
%AfterWindowOpening	4,000
%BrowserMethodCodeSection, '1', 'SetQueueRecord', '()'	4,000
%ControlPostEventHandling, '?JCO:JOBINTEREST1', 'Accepted'	4,000
%ControlPostEventHandling, '?TYP:JOB_TYPE', 'Accepted'	4,000
%ControlPreEventCaseHandling, '?CurrentTab'	4,000
%ControlPreEventHandling, '?Change:2', 'Accepted'	4,000
%ControlPreEventHandling, '?TagOne', 'Accepted'	4,000

```
If ~NumberTagged()
  Disable(?Print_Tagged)
Else
  Enable(?Print_Tagged)
End
If Cfg:Print_Limit
  If Mem:Number + NumberTagged() = Cfg:Print_Limit
    If Message('Maximum number of jobs selected ... Select OK to print tagged jobs, CANCEL to clear all
selections and start over.','Are you ready to print?',Icon:Question,Button:OK+Button:CANCEL) = Button:OK
      Pre_Print
      Post(Event:CloseWindow)
```

If an entire embed is to be deleted, this can be done by deleting the embed reference rather than the specific code. The code *can* be edited from the Change button.

Similarly, if none of the controls are needed for the template, all can be deleted in a single button press.

In large procedures, selective editing can get labor-intensive rather quickly.

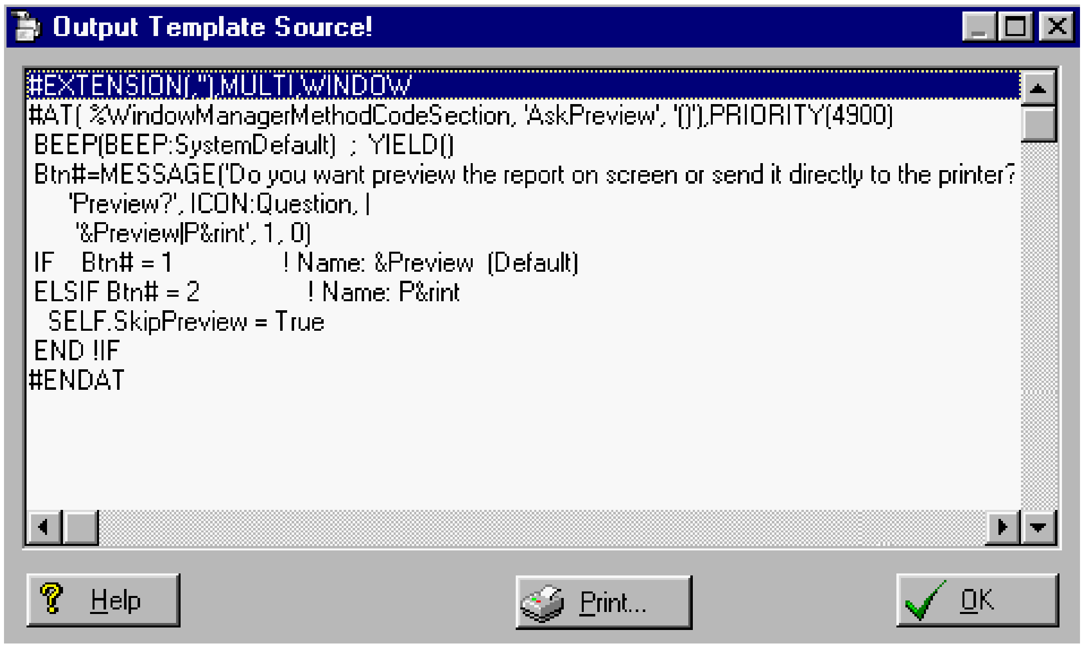
Rather than wade through all the code for a finished procedure, I decided to create a new procedure (actually a whole new app), containing only the custom code I wanted in the template. The resulting embed list (Figure 4, above) required no editing; all I had to do was remove the standard data and controls.

Overall, this approach strikes me as being much more efficient. In addition, by isolating only the code pertaining to the desired functionality, I can give it the attention it deserves (that is, I can clean it up.)

"Writing" a Template

Exporting to write the template results in Figure 5:

Figure 5. The generated template.



```
#EXTENSION(,''),MULTI,WINDOW
#AT( %WindowManagerMethodCodeSection, 'AskPreview', '()'),PRIORITY(4900)
 BEEP(BEEP:SystemDefault)  ;  YIELD()
 Btn#=MESSAGE('Do you want preview the report on screen or send it directly to the printer?
      'Preview?', ICON:Question, |
       '&Preview|P&rint', 1, 0)
 IF    Btn# = 1                 ! Name: &Preview  (Default)
 ELSIF Btn# = 2                 ! Name: P&rint
   SELF.SkipPreview = True
 END !IF
#ENDAT
```

As generated, the template is unusable for two reasons. First, it is a TPW, not a TPL. Either I have to create a TPL and `#INCLUDE('MYFILE.TPW')` this file or I must add a properly formed #TEMPLATE line as the first non-comment line (and rename the file to *whatever*.TPL).

Second, the template is not registered.

However the created TPL cannot be registered. It has a significant syntactical error.

Note the first line. A name and description are required as the first two arguments in any template declaration, including CODE and EXTENSION templates.

```
#EXTENSION(,''),MULTI,WINDOW
```

should be something like:

```
#EXTENSION(AskPreview,'Ask user to preview report at runtime'),↵
 MULTI,WINDOW
```

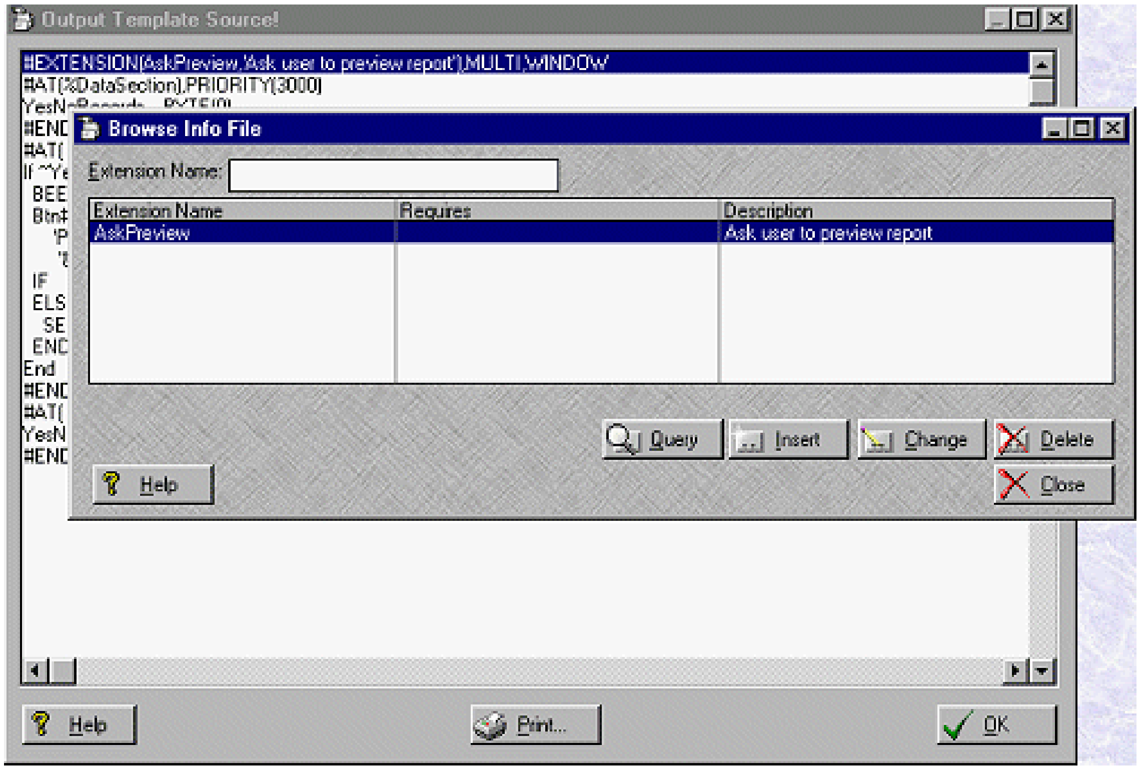
After that the template works.

> **NOTE:** When editing the TPW or TPL file directly, don't leave *any* blank lines; the template pre-processor doesn't like blank lines, doesn't like them at all.

As it turns out, if you want the Template Wizatron to plug in the name and description, the

Browse|Browse Info menu option can be tricked into doing this by simply leaving the "Requires" prompt blank:

Figure 6. Plugging in a template name and description.



```
#EXTENSION(AskPreview,'Ask user to preview report'),MULTI,WINDOW
#AT(%DataSection),PRIORITY(3000)
YesNoRecords    BYTE(0)
#END
#AT(
If ~Ye
  BEE
  Btn#
    'P
    '[
  IF
  ELS
   SE
  END
End
#END
#AT(
YesN
#END
```

Extension Name	Requires	Description
AskPreview		Ask user to preview report

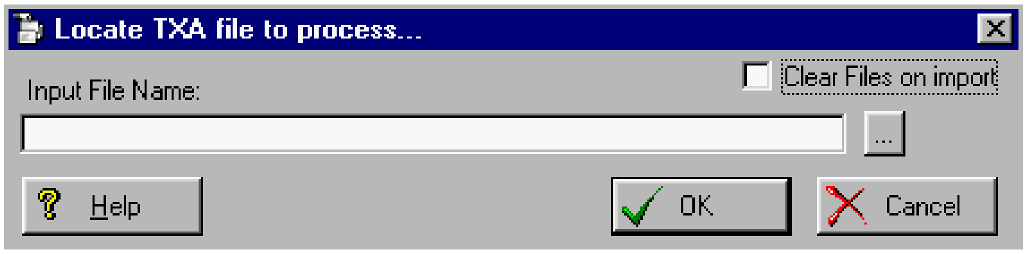
(This approach is also used to complete any instances of the `REQ()` attribute you may need. Count on needing some knowledge of the template language to use this feature.)

Good Stuff I Learned Along the Way

Without a doubt, the biggest caveat is the paucity of the documentation. Fortunately, trial and error harm nothing here (ego deflation excepted).

For example, it does not seem that you can reopen a previous TXA with edits applied. There is no "Open" prompt. However, the file import dialog:

Figure 7. Import TXA dialog.



has a checkbox, "Clear Files on Import." If you do not check it, your previous TXA, *with edits*, will be retrieved. This also means that a new TXA will be imported on top of the previous TXA. This allows combining templates (not always intentionally and not always a good idea but awfully handy when you need it as when building up a complex template from small pieces).

It also means that if you cancel this dialog, the previous workspace will be restored. So, you can, in fact, re-open a previous session. (The information used by the Template Wizatron is actually stored in a series of files in \C55\BIN: Controls.TPS, Data.TPS, Embeds.TPS, Info.TPS and Prompts.TPS.)

That the Wizatron does not automatically complete the name and description (or, since this *is* required information, prompt for it) is something to be aware of. So is the fact that the generated file is not automatically placed into a template chain. Neither are critical issues - if you fail to do either, you will find out quickly enough.

However, take another look at the header:

```
#EXTENSION(AskPreview,'Ask user to preview report at runtime'),↵
 MULTI,WINDOW
```

This template can be populated multiple times into any procedure and it also provides access to the window formatter. In fact, this extension is available in the application's global embed. This is not what is wanted (neither is it a major catastrophe either). Searching through the shipping templates and requisitioning code provides what is needed to fix this lack of elegance:

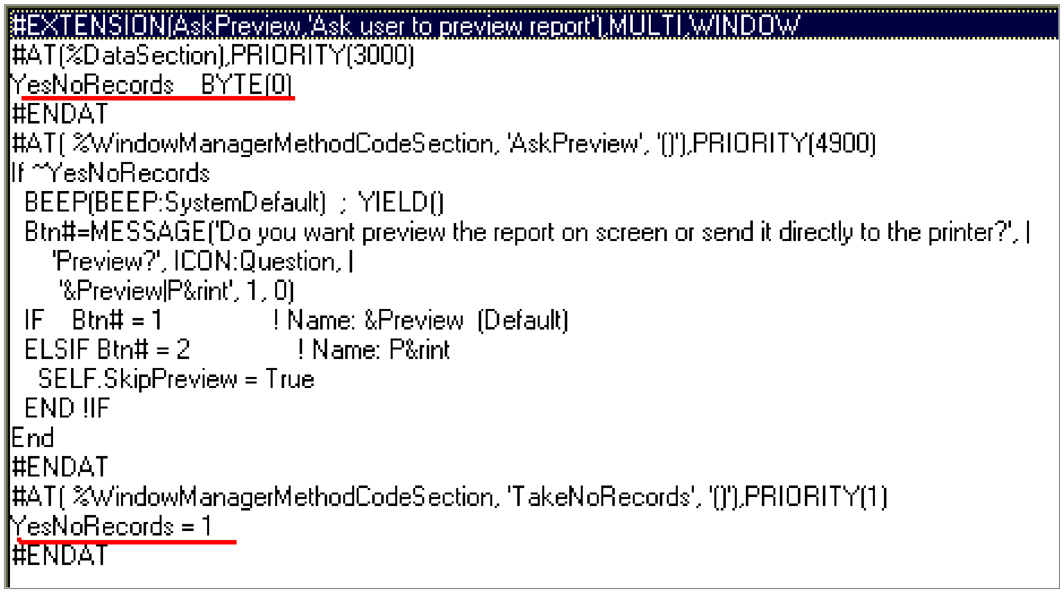
```
#EXTENSION(AskPreview,'Ask user to preview report at runtime'),↵
Procedure
#RESTRICT
#IF ( UPPER(%ProcedureTemplate) = 'REPORT')
  #ACCEPT
#ELSE
  #REJECT
#ENDIF
#ENDRESTRICT
```

If your template needs access to global embeds or embeds outside of the procedure proper, a hard choice is necessary. Either you'll need to add the required template code by hand or you'll need to create the TXA from the entire APP (makes my decision to create an APP strictly for creating a template seem almost prescient). Again, the shipping templates should provide sufficient examples and heuristics.

Hand work may well be required as you cannot edit the generated template in the Wizatron window (Figure 5).

By the way, after adding the variable I use to determine whether a report has records or not, the generated template looked like:

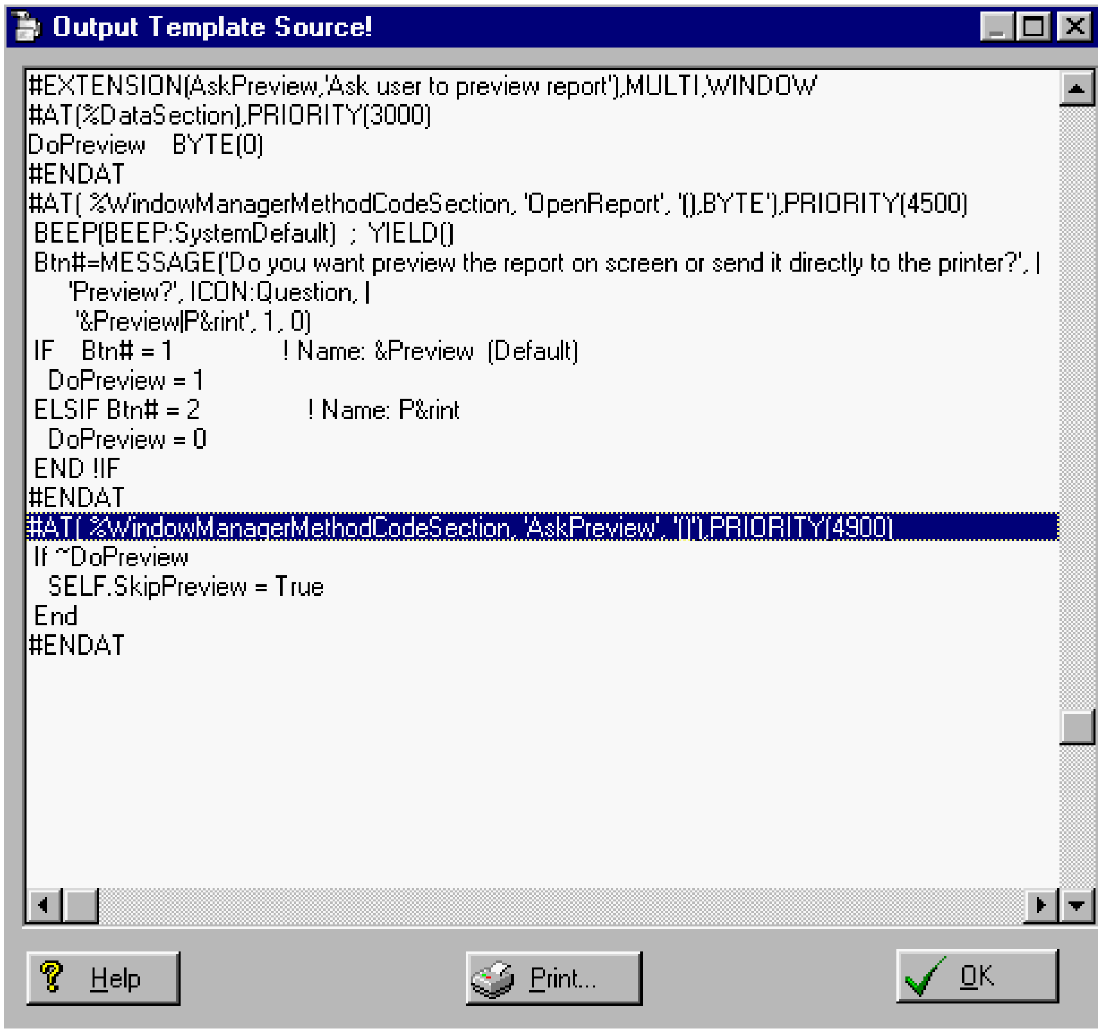
Figure 8. The final template.



```
#EXTENSION(AskPreview,'Ask user to preview report'),MULTI,WINDOW
#AT(%DataSection),PRIORITY(3000)
YesNoRecords    BYTE(0)
#ENDAT
#AT( %WindowManagerMethodCodeSection, 'AskPreview', '()'),PRIORITY(4900)
If ~YesNoRecords
  BEEP(BEEP:SystemDefault)  ;  YIELD()
  Btn#=MESSAGE('Do you want preview the report on screen or send it directly to the printer?', |
      'Preview?', ICON:Question, |
       '&Preview|P&rint', 1, 0)
  IF    Btn# = 1               ! Name: &Preview  (Default)
  ELSIF Btn# = 2               ! Name: P&rint
    SELF.SkipPreview = True
  END !IF
End
#ENDAT
#AT( %WindowManagerMethodCodeSection, 'TakeNoRecords', '()'),PRIORITY(1)
YesNoRecords = 1
#ENDAT
```

And, if you are a member of the "ask first" school of thought, it will look like:

Figure 9: "Ask first" template.



```
#EXTENSION(AskPreview,'Ask user to preview report'),MULTI,WINDOW
#AT(%DataSection),PRIORITY(3000)
DoPreview    BYTE(0)
#ENDAT
#AT( %WindowManagerMethodCodeSection, 'OpenReport', '(),BYTE'),PRIORITY(4500)
 BEEP(BEEP:SystemDefault)  ;  YIELD()
 Btn#=MESSAGE('Do you want preview the report on screen or send it directly to the printer?', |
      'Preview?', ICON:Question, |
       '&Preview|P&rint', 1, 0)
 IF    Btn# = 1                ! Name: &Preview  (Default)
   DoPreview = 1
 ELSIF Btn# = 2                ! Name: P&rint
   DoPreview = 0
 END !IF
#ENDAT
#AT( %WindowManagerMethodCodeSection, 'AskPreview', '()'),PRIORITY(4900)
 If ~DoPreview
   SELF.SkipPreview = True
 End
#ENDAT
```

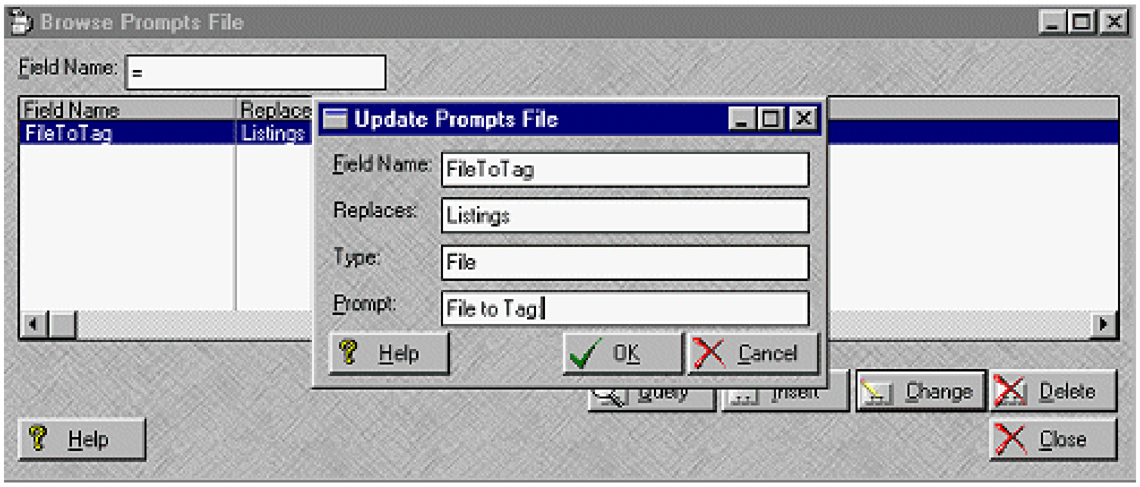
With a little bit of work and one additional template prompt, a single template could accommodate both quite handily.

Specific to General

Embedded code, of course, refers to the specific files and fields of the app, which hardly appropriate for a template.

Browse|Browse Prompts is used to convert field specific information into template prompts and code.

Figure 10. "Converting" a specific reference to a template prompt.



Field Name	Replaces
FileToTag	Listings

Field Name: FileToTag
Replaces: Listings
Type: File
Prompt: File to Tag

In this case, a template symbol, %FileToTag, will be created. It will be used to replace occurrences of "Listings" (a file in the dictionary). On the extension’s worksheet, the user will be presented with a prompt, "File To Tag:," and the IDE knows to expect a file label.

Unfortunately, there are no drop downs to aid selection of appropriate entries.

Summary

In general, larger, more complex extensions are likely to require considerable massaging. I tried exporting my tagging code. It was nowhere near ready to use, though it was much better than my last attempt to do it from scratch. I would estimate that perhaps a day, give or take, would be required to complete it. Since I do not use tagging all that often, I didn’t complete the template conversion (besides, there are commercial templates out there at a lot lower cost than a day of my time). My scratch attempt was at least an order of magnitude worse, so the Template Wizatron really was helpful.

The small template to prompt a user to preview a report took about an hour. That hour included learning my way through the Template Wizatron environment. Not bad.

I also did a template to replace the "No records" message in 10 minutes. Ok, 30 minutes if you include all the fancy formatting I added.

The bottom line? For code entirely contained within the embeds of a single procedure, small blocks of personal, custom code, the Template Wizatron is rather like Java: write once, use everywhere. Here the Template Wizatron is a clear winner. For more complex templates, where there is no third party template and you are not comfortable with the template language, the Template Wizatron is great kick start.

A word of warning: it isn’t perfect. The help needs expansion; drop downs for completing some of the prompts would be helpful; keyed storage of previous imports would be most helpful. But after your first success, TW.EXE gets addicting.

Steve Parker started his professional life as a Philosopher but now tries to imitate a Clarion developer. A former SCCA competitor, he has been known to adjust other competitors' right side mirrors - while on the track (but only while accelerating). Steve has been writing on Clarion since 1993.

For Immediate Release

October 17, 2000

COL Full Archive Available (beta)

Last week I referred to delays in publishing the COL archive, largely due to the articles having to be converted from PDF format. Several readers responded with HTML archives of their own, including Greg Miller and Alejandro Contreras, and I'm pleased to announce that the COL archive is now available online at http://www2.clarionmag.com/col.

So why are we calling the COL archive a beta? For starters, there are still a few missing images, and we haven't verified that all the source is included. Also there are some graphical inconsistencies that need to be cleared up.

The COL archive is also a beta because its host, www2.clarionmag.com, is a test server on which we're developing the next version of Clarion Magazine. Because of this the server may occasionally be unavailable. If this happens, just try again in a minute or two. If you experience lengthier delays or other problems, please email dharms@clarionmag.com with a description of the problem.

Dave Harms
Publisher

Web Development Options: An Overview

by Steve Parker

Part 1 of 2

If you're like most Clarion developers, you've at least considered the possibility of developing web applications. Since Clarion 2003, there's been out of the box support for web apps and it's not that difficult to hang something out on the Internet. Of course, life is hardly ever that simple. The problem is that Clarion developers, like other developers, face a bewildering array of web development options, some of which may not be immediately obvious.

Al Gore may not have actually invented the Internet but until he started talking about the "information superhighway," things were a lot simpler. The Internet was mostly confined to colleges and universities, almost unknown, and used the Telnet and Gopher protocols. The Web was in its infancy and browsers, such as there were, were primarily text based (remember Lynx?).

HTML, the *lingua franca* of the Web (HTTP is the transport protocol), supported text, emphasized text (italic in some browsers, bold in some browsers and nothing at all in others), strong text (sometimes bold) and, a bit later, images. The most complex structure was a <select> (the Web's equivalent of a File Loaded Drop box). For really fancy formatting you had <blockquote>, bulleted lists and <u> (underlined) and, for code and formulae, <pre>. Then image maps came along, two or three years after big Al started talking.

Andreeson left NCSA, took the Mosaic code created there and formed Netscape. Then there were Spry, Spyglass and a few others (remember "Internet in a Box"?). Two years later, Microsoft smelled the coffee. The race was well and truly on.

Now we have fonts, tables and style sheets, plug-ins, Javascript, applets, ASP and various plays on coffee words; it's a challenge just keeping up with the alphabet soup. All of this in the last four or five years. If Gore had just kept his mouth closed, the Web might be a whole lot easier to deal with (of course, then you wouldn't need this article).

And deal with it we must. Because, if nothing else, the hype has run roughshod over reality.

Despite the plethora of products making, or appearing to make, web development easy, scripts for the Web are complex and sophisticated. It is imperative that the typical Clarion developer comprehend the fact that moving to the Web is a change of platform and, because of that, there is a major learning curve. It just isn't as easy as anyone makes it sound.

Perhaps the most difficult concept to grasp is that, despite all the add-ons, plug-ins, drop-ins, turn-ons, tune-ins and drop-outs and whatever else has been created over the last several years, the Internet was not and is not designed as a platform for running applications. (Though, as it turns out, the ability to do so is a consequence, entirely unintended, of what it was designed to do. In other words, the Internet can be tricked.)

The Internet was created to allow transmission of messages from one terminal to another (computers are treated as terminals) across heterogeneous networks; it was created to move data. Originally, the Net was to move command and control data, later academic research results but in any case it simply moved information.

While networking is fundamental to the Internet, it wasn't the goal perused by its creators; TCP/IP was created as a necessary step to achieve the design requirement of allowing multiple existing networks to communicate with each other. TCP/IP is nothing more than a transmission protocol; it is not a network operating system. That is, resource sharing, in the way we are used to with PCs, is not an essential part of the Internet package. It's not part of the package at all.

Once given up by the Department of Defense and turned over to academia, the Internet quickly became essential for researchers sharing documents. With the advent of the Web came the proviso that any such document look substantially the same regardless of the computer on which it was being viewed or on which it had been created. The operative word is "substantially."

Too many Clarion developers think that the Internet is a sort of magic pill for their applications. "It works in Windows," they think, is sufficient to ensure that the application will work on the Web, without modification no less. Many continue to believe that their applications will operate on the Web just as they do in Windows, that the Internet is a network in the same sense as a LAN. The only certainty, in fact, is that if it doesn't work in Windows, it won't work on the Web either.

A sign over the National Archives in Washington, D.C. says "The past is prolog." Before you actually bring your apps to the web, do your research. Read the history of the Internet. This exercise may take as much as half an hour but will teach you the underlying nature of the platform. Do an HTML tutorial or two.

Get a feel for the nature of the beast. Learn what it does and what it can be made to do. That is, get familiar with its warts.

Without a knowledge of the platform, your Web experience will be more frustrating than it needs to be (and the limitations of the platform make it frustrating enough to begin with).

Remember, a Web page is supposed to display, with *approximately* the same look, on any computer. This means PCs, Macs, Sun, Amigas or another computer capable of connecting to the Web. Now, as a Clarion developer, you develop for Windows; why would you think your Windows apps will run on Macs or AIX boxes? Why, then, would you expect your Windows app to migrate to the Web without change?

The single most important Clarion element you lose going to the Web is the Accept loop: a browser doesn't have one (and there is nothing you can do about that). By extension, this means that you lose your embedded code. Because embedded code is executed by the Accept loop, it will not execute as expected (Clarion does provide built in options to permit embedded code to execute as substantially expected, but that's another story).

So, let's assume you've done your homework, you understand the limitations of HTML and HTTP, and you still want to put your application on the web. What are the alternatives for a Clarion developer?

What follows is not an exhaustive list of the ways a Clarion developer can create a database application for the Web. It is a summary of the most popular ways Clarion developers do web development.

Doing The Web Thing

I have gone out of my way, above and in various articles on Internet Connect and Web Builder, to point out that applications on the Web do not and, I have claimed, cannot behave exactly like their Windows counterparts. Now I have to admit that this is not entirely true (and I've known it all along). There is a way to make your Windows apps behave exactly the same on the Internet as they do in Windows.

Fat Client Technology

To make your apps run more or less unchanged on the Internet you do not use "thin" client technology (if a 50 meg browser can be considered "thin"). Instead, your EXE and required DLLs are installed on the end-user's PC. The "magic" comes in using SQL and the Internet as your network (remember, the Internet can be tricked).

Because SQL uses Data Source Names (DSNs) to connect to databases, if your SQL database is on a machine with a fixed IP address, you can have the user set up a DSN pointing to your database (firewalls permitting).

Mike Pasley describes this technique as a way to create three tiered Internet Connect applications:

> It is usually simple to implement a basic distributed and three tiered system with CWIC.
>
> [With] SQL, you can build an application where CWIC is on one server and the data is on another server using a SQL client (MS SQL, Oracle, Pervasive, Adaptive, etc. ... your choice) and a driver. (The data source, once set up, handles the IP connection.)
>
> The client will point to the data server through an IP address (or domain name). You will have a connection with the user id, password, and database name that you've set up in your app. It will correspond to the user id and password for the database you set up in the database server. Your ODBC driver or native driver (I've used the Oracle Accelerator from TS) will communicate with the db server through the client. You can do this in Clarion so you can do this in IC.

So your data can be anywhere on the Internet.

As Mike correctly notes, "you can do this in Clarion." So, you can run your apps using the Internet as your network. Of course, you are limited to running on Windows PCs or PCs that can execute Windows' binaries. And, you are not using a browser (not a great loss, in my opinion).

If you are already using (or migrating to) SQL and you require the security that comes from the ability to restrict access to your app (it has to be installed on the client PC) and/or you need the total control that comes from being able to use a single platform, this method of publishing an app must be seriously considered.

Remote databases aren't necessarily slow. At ETC 2000 Dave Harms demonstrated a Clarion application using a MySQL database running on a Linux server in his Winnipeg office. Despite being limited by a 28.8 dialup connection, performance was snappy.

The Internet Framework Templates

Mike Pasley's Internet Framework Templates (IFT) were the first major third party Internet related Clarion add-on.

IFT are unique in that they don't they require the Application Broker. In fact, they don't even require a web server (IIS, Apache, O'Reilly's or the like) and the headaches that go along with installing, configuring and maintaining each.

Adding IFT to a Clarion APP turns the application itself into a web server. This means that the Clarion binary is able to listen for and respond to HTTP requests *without* additional software. This makes it possible to deploy an app to a PC with a fixed IP address and have the app instantly on the Web (the PC is an instant server). It also means that if the server PC has Web Server software running, your app must be configured to listen on an alternate port (80 is the default for HTTP).

While Mike has an additional product that will create basic HTML for a browse/form combination, this capability is not built into the base product. IFT normally requires the developer to supply HTML for each procedure. Using a visual HTML editor and place holders for your database fields, this is not especially difficult but it is not technically RAD. However, IFT results in extremely fast and stable page delivery.

Clarionet

A template set created by Michael Brooks, Clarionet was created because his application windows (structural engineering applications) were too complex and computationally intensive for decent presentation or performance using standard approaches.

With Clarionet, the remote client sends a command to the Clarion Application Broker and the server passes the commands on to the app. The app starts and the Clarionet server app scans the screen and sends it to the client. The server then waits for further commands (requests) from the client (in other words, it behaves like a *daemon*).

User actions are sent to the server and mirrored on the app running on the server. Buttons clicked or keys pressed on the client are passed to the server (the app) and the action mirrored there.

So far, this sounds substantially the way a standard Clarion Internet Connect (IC) or Web Builder (WB) app works. The difference between IC, WB and Clarionet is what happens when the server app is done processing user events. In Clarionet, when the window is ready for redisplay, Clarionet re-scans the screen and sends the changes to the client (and, note, it is send a representation of the Window, not an HTML translation). That is, it sends the changes *only*. IC and WB apps re-send the entire page.

In other words, a Clarionet-extended app runs *as if* it were running on the client. It looks like a Windows app.

Because of the remote emulation, Clarionet does not support API calls, timer-based events or OCXs. It also requires a one time download of a 600k EXE (so it can appear to the end user to be a plug-in).

Clarionet was designed as a lower cost alternative to Citrix and Terminal Server (see below) and has earned an excellent reputation. When proprietary apps are to be deployed and the increased level of security and control of staying within Windows are your goals but SQL is not necessarily required, Clarionet deserves your attention.

Citrix

Probably the first product to capitalize on the network underlying the Internet, Citrix provides a framework for running apps as if on a LAN. And, you will pay for this; Citrix can become very expensive very quickly.

Requiring a one-time download of a client, Citrix allows a high level of security and is reported to be quite quick when configured properly (unfortunately, my limited experience running apps in Citrix does not confirm any performance claims). Clarion apps from CPD 2.1 through Windows 5 are reported to run on Citrix without problem.

Citrix handles head down data entry well but does not suit general access applications. Instead, Citrix is design for private access. Developers using Citrix report that it supports 50-100 simultaneous users per box, depending on app and data space size.

Citrix, then, is a strong solution for highly secured apps, remote data entry; internal corporate apps.

Terminal Server

Terminal Server is Microsoft's version of Citrix. While it starts out inexpensive (it is included with 2000 Advance Server), its cost can escalate quickly (more quickly than Citrix).

Reports from Clarion developers indicate that, at this time at least, it is not entirely stable and that, for the purpose, Citrix is a better choice. Do note that if you deploy an application for access with Terminal Server, it must be either a standard windows app or it must be configured for dual-mode operation.

ASP

Microsoft's Active Server Pages are an entirely non-Clarion solution to publishing a database on the Web. And, while ASP began as a proprietary Microsoft solution, there now exist servers, like ChiliSoft, to run ASP scripts on non-NT servers.

To use ASP, you must also use ODBC as your file interface layer. This means either SQL for your database or using the SoftVelocity ODBC driver for TPS files.

There are a number of visual tools for creating ASP. FrontPage 2000 and Drumbeat (recently purchase by Macromedia, makers of Dreamweaver, and renamed Dreamweaver

UltraDev) are probably the best known. The WYSIWYG features may not always be two way, however. That is, after ASP scripting is added, you may loose WYSIWYG in FrontPage (I have one report of this).

ASP products currently on the market are good for medium sized sites, 50-100 simultaneous users under NT. Reports are that current NT implementations do not scale well beyond 100 or so users and scripts are prone to erroring out particularly during db access (at least this is true of the ASP site I visit most often, leading me to question the stability of the ODBC interface). ASP sites also tend to be slow but this is likely less due to the speed of executing a script than the number of images used on the typical page (everyone knows that a record form simply *must* have seven or eight graphics).

ASP running under Windows 2000 Advanced server is reported to be *fast*. Using Microsoft’s stress tool, a turnkey e-commerce package using ASP ran over 100,000 users at less than 20% memory utilization (128 Mb).

During the recent posting of 5.5 Release Candidate 1, an interesting remark was made by a SoftVelocity employee. It seems that SoftVelocity is beginning work on an ASP template set. Nothing is known beyond the fact that a version was used to create part of the CR1 site and that we weren’t to ask further questions at this time.

Another fact: we know that the site built with this template set was able to max the ISP’s pre-set load limits. The ISP puked, not the app.

Given Clarion’s built in data access, there exists a real possibility that this technology will open to us in the future and with somewhat greater stability than the typical ASP site.

Next week I’ll finish up with a look at SoftVelocity’s web development offerings, and some non-Clarion options.

Steve Parker started his professional life as a Philosopher but now tries to imitate a Clarion developer. A former SCCA competitor, he has been known to adjust other competitors' right side mirrors - while on the track (but only while accelerating). Steve has been writing on Clarion since 1993.

Press Release: COL Full Archive Available (beta)
(Oct 17,2000)

Web Development Options: An Overview - Part 1
(Oct 17,2000)

An Introduction To Writing Templates: Part 2
(Oct 17,2000)

October 2000 News
(Oct 17,2000)

An Introduction To Writing Templates

by John Morter

Part 2 of 2

In Part 1 of this article I laid out some Clarion template language basics, and I explained my borrow-and-adapt approach to template writing which led to the *borrowing* of ideas (and template code) from the standard Clarion `BrowseSelectButton` and `FieldLookupButton` templates. Now it’s time to further *adapt* this framework to create a new template that executes some logic when a button is pressed.

Borrow-And-Adapt, Continued

In trying to keep things simple, I’ve been purposefully vague about the details of this sample template thus far, but the original request I saw on the newsgroup was very specific. The requirement was to invoke the following pseudocode when the button is pressed:

Listing 1. Applying form updates

```
! Apply Form updates
IF ThisWindow.Request = InsertRecord
  Access:<Filename>.Insert()
  ThisWindow.Request = ChangeRecord
ELSE
  Access:<Filename>.Update()
END
```

The idea behind this request is to put an "Apply" button on a Form that allows the pending action defined by `ThisWindow.Request` to be applied.

I’m using this example (without commenting on the code itself) because it is typical of the sort of thing you want to do when a button is pressed (that is, to execute some code), and because it presents the extra challenge of needing to convert the generic idea of a <*filename*> into a specific reference to a valid filename.

After the `#CONTROL` statement the actual control (or set of controls) is declared. Control(s) defined here will be "attached" to the mouse-pointer for placement on the window by the developer. And now that I’ve explained the intent of this template, I can be specific about the required control:

Listing 2. The button control template

```
CONTROLS
  BUTTON('Apply'),AT(,,45,14),USE(?ApplyButton),↵
    TIP('Apply updates to database'),SKIP
END
```

You won’t need any help understanding this - it’s all standard Clarion code.

User Involvement

As I mentioned in the introduction to Part 1, one of the great things about templates is that they’re polymorphic – one template can generate quite different code for different circumstances. This is achieved by capturing different settings or choices via the template-to-developer interface - and the key to this is the `#PROMPT` statement.

In this specific example, the actual filename to be updated by the *Apply* button must be determined so it can replace the <*filename*> placeholder in the pseudocode shown above. Here’s where template symbols come in.

There are two distinct types of template symbol: Those that are user-defined, just like your own variables in Clarion code, and those that are built-in, just like built-in functions in the Clarion language.

> **Tip**: There’s also a subtle third type (but this is just my personal interpretation, not one I’ve seen mentioned in the PG): there are symbols defined by the standard shipping templates which *could* be really useful to have access to - if only one could get access to them! More on this, and built-in symbols, later.

Since the name of the file to be updated is needed, the developer will be asked for it, and the response given will be stored in a user-defined template symbol - say, `%UpdateFile`:

```
#BOXED('Apply Button properties)
  #PROMPT('File to be updated',FILE),%UpdateFile,REQ
#ENDBOXED
```

The `#BOXED ... #ENDBOXED` structure just puts a nice box around the template properties prompt - it’s not required, but it does look good!

The `#PROMPT` statement is where all the input action happens. A `#PROMPT` asks the developer for the name of the file to be updated. `FILE` is a key word that forces selection to be one of the files in the data dictionary. This file is chosen from the file schematic for the current procedure. The `REQ` attribute makes this a mandatory entry, and `%UpdateFile` will end up holding the name of the specified file.

At "Here", Embed This ...

All that’s needed now is to define the code to be embedded along with the control and you’re done! (Unfortunately, it’s actually not quite as simple as that, as you will soon see.)

Listing 3. Embedding code

```
#AT(%ControlEventHandling,%FormApplyButton,'Accepted')
! Apply Form updates
IF ThisWindow.Request = InsertRecord
  Access:%UpdateFile.Insert()
  ThisWindow.Request = ChangeRecord
ELSE
  Access:%UpdateFile.Update()
END
#ENDAT
```

The `#AT(...)` structure tells the generation process where to embed your code. However, working out exactly what to specify for the `#AT` is often one of the most difficult parts of template writing. I usually revert to my borrow-and-adapt approach.

Examining the `FieldLookupButton #CONTROL` template, it's not too hard to work out that its `#AT(...)` statement defines an embed point during Control Event Handling where the control is accepted. That's exactly what is needed for this example, so I just borrowed the same code and adapted it to suit. At this stage you should be wondering about the `%FormApplyButton`, but don't fret, I'll explain it shortly.

So within the `#AT ... #ENDAT` structure you can simply slot in your code. In this specific example, the %UpdateFile template symbol will be replaced at code generation time by a real filename.

> **Tip**: For details of other options and attributes on the `#AT` statement, see the Programmer's Guide (PG). For instance, you might need to consider the `PRIORITY` attribute to get your code placement exactly as you want it. Also check out the many other built-in template symbols available to you.

Fine Tuning

Have you noticed when using the `FieldLookupButton` control template that it's easy to blithely populate the control on your window and then exit the Window Formatter without filling in the Field Lookup Button Prompts? And then you get error messages, such as "*File Lookup needs to refer to Entry Control*", during code generation stage!

Examining the `FieldLookupButton #CONTROL` template again, you'll notice this error message as part of the `#ATSTART … #ENDAT` structure.

This is a good example of defensive (template) programming, to ensure that certain conditions which the template code depends upon are properly satisfied.

According to the Template Language help-text, "the `#ATSTART` structure specifies template code to execute before the `#PROCEDURE, #CODE, #CONTROL`, or `#EXTENSION` generates its code." It kicks in just before code is generated, so it's the ideal place to check for property completeness.

Following my borrow-and-adapt approach results in:

Listing 4. Checking for property completeness

```
#ATSTART
  #IF(NOT %UpdateFile)
    #ERROR(%Procedure & ' Error: No filename specified ↵
      for Apply button')
  #ENDIF
#ENDAT
```

In Listing 4 `%Procedure` is a built-in symbol that usefully identifies the procedure with the template error.

If you were to put this template code together now and try to register it you'd hit an error on the

#AT statement - because the `%FormApplyButton` symbol has not been declared.

Why do you need this symbol? Because, without it you have two problems: Firstly, the `#AT(...)` statement cannot refer to a specific, "real" field-equate-label (such as `?ApplyButton`) because the template registration process will not recognise it. And secondly, even if it could, you cannot guarantee the developer will never change the USE attribute of the control after he's populated it with your control template - even if you're writing the template only for your own use!

It's much better to make the template truly generic. Here's how:

You need some template code that will identify the field equate label for the control you're populating with your template. In pseudo code terms: scan through a list of all field equate labels for all controls on the current window and find the one that corresponds to the control that was populated by this `#CONTROL` template.

You'll find variations on this approach in just about every `#CONTROL` template. I borrowed the solution from the `BrowseSelectButton #CONTROL` template;

Listing 5. Identifying the field equate

```
#ATSTART
  #DECLARE(%FormApplyButton)
  #FOR(%Control),WHERE(%ActiveTemplateInstance=%ControlInstance)
    #SET(%FormApplyButton,%Control)
  #ENDFOR
#ENDAT
```

> **Note:** The template language concepts involved in this step are more advanced than covered so far, but don't be put-off. My borrow-and-adapt approach requires only that you recognize likely code-candidates for the problem facing you, not that you could whip off a quick solution all by yourself. More often than not it's a "try it and see" process, ideally assisted by a good text-searching tool.

A bit more template language theory is needed before jumping into this code: Some template symbols are multi-valued - they can be thought of in much the same way as an "array". `%Control` is an example of a multi-valued symbol, and it's also a built-in symbol. It holds the field equate labels of all controls on the window (just what's needed!).

> **Tip**: Check out these template language statements in the PG; constructs like this are typical.

Both `%ActiveTemplateInstance` and `%ControlInstance` are built-in symbols.

- *`%ActiveTemplateInstance`* is a multi-valued built-in symbol which contains the identification numbers (instance numbers) internally assigned by the template "engine" to all control templates in the procedure. This template is going to be one of those.
- `%ControlInstance` is a single-valued built-in symbol which contains the identification number (instance number) internally assigned by the template "engine" to the control template which populated a particular control onto the window. That's us!

Armed with this information, the template code is not too hard to follow:

First, the `%FormApplyButton` template symbol is declared.

```
#DECLARE(%FormApplyButton)
```

Then, `%Control` is pinned-down to the specific array-element corresponding to the control `where` the template ID-number matches the ID-number of the template that populated that control. After which you can `set` the contents of `%FormApplyButton` to the field equate label held in that array-element (whatever it may currently be).

```
#FOR(%Control),WHERE(%ActiveTemplateInstance=%ControlInstance)
  #SET(%FormApplyButton,%Control)
#ENDFOR
```

That’s it! The template now does exactly what was asked of it. See below to download the complete template source code. Of course you may not have an urgent need for an Apply button template, but you should now be able to borrow-and-adapt from it to build a template that suits your own specific needs.

Wrapping It All Up

When you download the template you’ll find two different file types. *AB_Addon.TPL* is a template chain header, (identified by its `#TEMPLATE` statement) - I use this file as a "container" for my own template collection, as distinct from the standard shipping templates.

jmApply.TPW is the template discussed in this article. I’ve also included a couple of other simple templates for you to explore.

To use the template, move all the .TP* files into your *Template* directory, then select *Template Registry* from the *Setup* menu and register *AB_Addon.TPL*.

Wait, There’s More!

Another posting on one of the newsgroups caught my eye, because the question was one that I had once puzzled over myself: someone asked whether there was a way to have access, from within a template, to a list of all files used in the application. At first glance, the answer seemed to be via the built-in `%File` multi-valued template symbol, but `%File` contains all files in the dictionary, whether or not they are used in the application.

I watched this thread with interest. The originator came up with his own solution that involved some clever (but roundabout) template coding - and then invited suggestions for improvements.

There was one aspect of his solution that jumped out at me. It referred to a multi-valued template symbol (`%UsedFile`), declared and used in *ABChain.TPL*, that seemed to provide the exact answer we were both looking for - yet access to this symbol was not available from within our own templates. It got me wondering…

I’ll take you through the process I followed to solve this mystery because it further demonstrates my borrow-and-adapt approach. (*Note: I later found out that the answer was well known to the template-writing old hands - but I didn’t know that then!*)

First, I scanned through *ABChain.TPL* and had a good look at how `%UsedFile` was being used. Sure enough, it was just what I wanted.

So, I created a simple test template and tried it out. The template "compiler" didn’t complain about me referring to `%UsedFile`, even though I had not declared it - but it was empty.

Next, I wondered if any of the other standard templates were making use of this symbol. I pointed my text-search tool at *AB*.TPL* and got quite a few "hits".

In one of the simpler templates, *SetABCProperty*, I could see that `%UsedFile` was clearly not being (explicitly) declared - and yet the template was happily referring it to. This definitely warranted some digging into.

I noticed that this template has a #PREPARE structure as follows:

Listing 6. Using #PREPARE to populate a multi-valued symbol

```
#PREPARE
#CALL(%ReadABCFiles)
#CALL(%BuildObjectList,1)
#IF(~%ObjectName)
#CLEAR(%PropertyToSet)
#ENDIF
#ENDPREPARE
```

The `#CALL(%ReadABCFiles)` looked promising - so I tried it in my test-template. To my delight (and surprise, I must admit) `%UsedFile` sprang into life and gave me an "array" of files only used by the application. Wow! It's not very often I get lucky first time!!

As you can see from this example, you can discover a lot by exploring templates,- and once again a good text-searching tool is indispensable.

For your interest, here's my test template:

Listing 7. The test template

```
#!================================================================
#EXTENSION(TestUsedFile,'Test UsedFile symbol'),APPLICATION
#!
#DISPLAY('Check the contents of AppName.clw after generation')
#!----------------------------------------------------------------
#ATSTART
#CALL(%ReadABCFiles(ABC)) #! ReadABCFiles is in the (ABC) chain
#ENDAT
#!----------------------------------------------------------------
#AT(%BeforeFileDeclarations)
#!
#FOR(%File)
! Dct file = %File
#ENDFOR
#!
#FOR(%UsedFile)
! Used file = %UsedFile
#ENDFOR
#!
#ENDAT
```

For more template writing tips, and some simple examples to explore, check out the Sterling Data template pages. As well as the standard shipping templates, other good sources of examples to learn from are the many public domain template sets such as those from Locus Software, Mike Hanson and Mike Pasley's *Simplates*. Also have a look at the excellent Clarion resource links at the TopSpeed Turnpike and at the Clarion Magazine categorized links page.

Download the source

John Morter is a member of the Victorian Clarion Users Group (Melbourne, Australia). His moneymaking day job doesn't actually involve Clarion (at least not officially), but Clarion occupies a lot of his spare time as a hobby to keep his techo-developer background up to date. He sails in the bay during the summer on his racing catamaran named Flat Chat, which is Australian slang for "at top speed" - or "at high velocity".

Clarion MAGAZINE

Web Development Options: An Overview

by Steve Parker

Part 2 of 2

Last week I began an overview of web development options available to Clarion developers. This week I'll conclude with a look at SoftVelocity's offerings, and some non-Clarion options.

SoftVelocity Solutions

ASP rumors aside, SoftVelocity provides three different methods of taking an app to the Web: Internet Connect, Java-free Internet Connect (unofficial), and Web Builder. All can handle 200 -250 simultaneous users (of course, this assumes no queues are left unfreed and all loops have an escape hatch).

In all cases, applications are deployed on an IIS compliant web server and uses an application broker. The broker serves HTML (and optionally JavaScript) created by the Internet Connect or Web Builder classes to reproduce the application's functionality. The broker delivers this HTML (and optionally Java classes) and responds to information sent back from the client (as when a form is completed or a browse is paged).

Internet Connect was introduced as CWIC and has evolved into the template set still shipping with 5.5. Web Builder is the legacy of the Wizatrons, new with 5.5.

Internet Connect, Java-free

Internet Connect was heavily modified early on by Tony Goldstein. Those modification are significant enough that they actually comprise a third, though not out-of-the-box, alternative: Internet Connect, Java-free (templates based on Tony's work, supplemented by Arnor Baldvinsson and myself are available at the end of this article).

Applications using these modified IC templates run entirely without Java or Javascript (Web Builder uses no Java but does rely heavily on Javascript). This is easily proven by turning off both Java and Javascript (quite easy to do in Netscape) and running an IC

Java-free app. It will run without difference; the CWICWEB download app is an example.

There are, however, substantial compromises to be made when using Java-free apps.

(1) Because there is no Java or Javascript, the standard Clarion options allowing a page refresh are not available ("Refresh when changed " in Web Builder and both "Full' and "Partial Refresh" in IC). These options are used to trigger embedded code and, without them, the developer must do an explicit Submit to force the issue. (Note the "Apply" button in the example app.)

(2) HTML list boxes only support a single string (column). This is the nature of the HTML `<select>` which is the only list ("list" in the sense that Clarion developers understand) HTML supports.

Fortunately, you can use the browse queue provided by the standard templates, suppress normal HTML generation for the list and format your own HTML for the list area. The details on doing this have been previously published; check the reincarnation of Clarion Online here at Clarion Magazine for "Decaffeinating Forms and Browses," "Dual-Dual Mode Apps," "Dual Hybrid Apps – The Details" and "The Best of Both Worlds: Browse Boxes on the Web."

Why go to this kind of effort? Well, in the first place, it really isn't as hard as it sounds (especially after mastering Vince Russo's technique as presented in "The Best of Both Worlds" – the CWICWEB app uses a variant on this). Second, if an app must support every browser ever made, this is the only Clarion solution. Third, with only a little effort, apps look *really* good.

Internet Connect, With Java

The original CWIC, Internet Connect still ships with 5.5. It is included not simply to support apps created in earlier version but because it is still a viable solution for creating web database apps.

Looking at a standard IC app, it seems clear that emulating windows applications was a design parameter for this product. Browse boxes and thumbs behave just as the do in Windows, incremental locators do also. Default buttons respond to the Enter key.

Because such apps look and operate very much like apps deployed in Windows or even on Macs, training issues are minimal. On a company intranet, with 100 Mb hubs and NICs so common and, externally, broadband Internet access so common, IC with Java should not be dismissed out of hand.

Coding these apps is closest to coding Windows apps. Scroll bars actually scroll and these apps are inherently dual mode (allowing Windows clients to access them across a LAN and

Mac or Unix clients to use a browser). In short, IC is just the easiest way to take a Clarion app onto the Web. When bandwidth is not an issue, IC imposes the fewest compromises.

Web Builder

The latest and greatest from SoftVelocity (at least until the ASP templates come), Web Builder uses no Java. WB apps generate pure HTML with Javascript embedded in the

Skeletons.

The "Skeletons" are new. These are a series of files, one per control, to create HTML for a control from information passed by the app to the Web Builder classes.

Like ASP plus ODBC, the presentation layer, contained in the Skeletons, is almost entirely separated from the business layer. With ASP, you embed OBDC calls. Similarly, the Clarion app contains not only the file access (like ASP + ODBC) but also data validation and any desired calculations/post processing (similar to SQL stored procedures). The Skeletons contain pseudo-HTML structuring what the end user will see. Though HTML can still be embedded in the app which will affect the page as seen on the browser, the vast majority of the look and feel is controlled by modifying the Skeletons.

The notion was that the Clarion developer would do the app and turn the Skeletons over to an HTML specialist. This architecture has the advantage that Skeletons can be changed without recompiling the app. Unfortunately, the real effect has been to force Clarion developers to learn more HTML than most would prefer. On the other hand, once you've come to terms with the architecture, it *is* quite flexible. But if you'd rather just name background images and colors in template prompts, IC may be your cup of tea.

One of the most significant additions with Web Builder is the "linked in broker." This feature allows you to press the run button and test your app in a browser. By judicious copying of files to floppy, you can distribute preview-in-a-browser demos of your apps.

Integrating With A Web Server

There are a number of other ways of taking your apps onto the Web. Hand coders may want to consider CGI, the Common Gateway Interface. CGI is a standard by which a web server can call an external program and generates HTML that is incorporated into a web page.

Many CGI programs are written in Perl and C, but you can easily write CGIs in Clarion as well. Information is passed from the server to the CGI program using environment variables, or in the case of the Win-CGI variant, via INI files. The CGI program reads the passed data and generates the appropriate HTML. If you borrow the basic HTML from a visual formatter, this is only difficult the first time (but according to Skip Williams, the Clarion guru in this area, that "first" time can be on the painful side).

The downside to CGI is that each time the web server processes a CGI request, the CGI program has to be loaded and run. On a busy site CGIs can take a lot of processor time and memory. To solve this problem, you may want to consider a solution that's more tightly integrated with the server.

For example, the Tornado OCX makes it relatively easy for your Clarion application to talk to the IIS web server through the IISAPI. This way your application is loaded once, and receives and responds to requests via the OCX. Tornado is well liked by those who use it but unfortunately is no longer officially supported.

Non-Clarion Web Development

Quite a lot of Clarion-related web development is supplemented by other languages such as Java and JavaScript. You can take things one step further and use an entirely non-Clarion approach for the web part of your application. So is this Clarion development at all? Well, you’ll probably still find it easier to design the database using Clarion, and often there’s a role for a database maintenance application to supplement the web functionality.

Many Clarion users like the Cold Fusion products which allows pixel level placement of page elements. Well constructed Fusion sites scale well, and are able to handle quite large sites.

ASP, mentioned earlier, can fall into this category as well. And you can even write Java servlet- and JSP-based web applications which use a database otherwise maintained with a Clarion application.

Which Is For You?

There are no magic pills here, no recommendations. There are Clarion developers using each of these methods, happily and successfully. Among the SoftVelocity-supplied solutions, each has place and a purpose. An import point is the realization that some types of apps simply don’t belong on the Web; proprietary, high security, data entry intensive app are among them. But this does not bar you from using the Internet to deploy them.

Which approach is best? The one that works for you.

Of course.

Download the source code

Steve Parker started his professional life as a Philosopher but now tries to imitate a Clarion developer. A former SCCA competitor, he has been known to adjust other competitors' right side mirrors - while on the track (but only while accelerating). Steve has been writing on Clarion since 1993.

Reborn Free CLARION online

published by CoveComm Inc.

Clarion MAGAZINE

Main Page

COL Archive

Log In
Subscribe
Renewals

Frequently Asked Questions

Site Index
Article Index
Author Index
Links To Other Sites

Downloads
Open Source Project
Issues in PDF Format
Free Software

Advertising

Contact Us

Feature Interview: Mark Riffey

Web Development Options: An Overview - Part 2
(Oct 24,2000)

Feature Interview: Mark Riffey
(Oct 24,2000)

Mark Riffey is President and founder of Granite Bear Development. Granite Bear's Photo One studio management program has been named the HOT 1 award winner by the trade journal of the Professional Photographers of America. Mark spoke with Dave Harms, Clarion Magazine's editor and publisher.

Tell me a little bit about Photo One. It handles all aspects of photo studio management?

Basically it goes from the front of the studio to the back of the studio. The way a studio is laid out, they have a reception area where the appointments are made, they have video presentation or still presentation rooms, they have sales rooms, they have camera rooms where the shots are done, and then all the back office production areas. The program handles each one of those areas.

What got you started in this market? You're a photographer yourself, I take it.

Well, kind of a rookie photographer. My Dad worked for Kodak for 25 years. But actually I met a guy in the Compuserve Clarion forum, Stuart MacFadden, and he lived in Massachusetts. He'd been working with this CPD program as a consulting project for a studio up there, and he arranged to sell it on his own. In 1998, after spending quite a while working on the Windows version, I think he needed a change. He went into the insurance business, and we just ended up striking a deal. He had a sales person working for him whose family has a studio, so she knew the business inside out. While I knew my way around a darkroom, I didn't know squat about what happened at a portrait studio. She was kind of a dealbreaker for the thing. "If she doesn't come, I'm not going anywhere." She came along with the deal and she's still here.

Is your product specific to the US?

It's mostly specific to the US and Canada. We've had a little bit of interest from Mexico, we've got a couple of users in Australia, we've got a distributor that's just getting off the ground in New Zealand. But beyond that we haven't really put a lot of effort into internationalization. There's been some discussion about moving into Europe but we don't know how the studios work over there.

I imagine there could be a lot of differences.

Even between the US and Canada there's a big difference in the high school senior business. Our program is the only one that has any high school senior features, and there's a ton of them in there. It manages the process of keeping in touch with the schools, contact information; for every yearbook there's a number of different things you have to keep track of. These days there are even rules in the contracts that kids can't wear hats, you can't be showing anyone's hands, it's unbelievable all the little details you have to keep track of.

Are yearbooks a major part of the business?

Workflow is a big part of it. Anybody can use QuickBooks to manage their sales and their financials, but the workflow and the marketing side really sell the product.

Is the move to digital happening, and will that change how your software is developed?

No, not really. The big move hasn't happened yet, but I think we're looking at seeing it in the next couple of years. There are a couple of new cameras coming out, and there was a new 16 meg CCD coming out that's actually reasonably priced. Right now the technology behind the digital cameras can't keep up with film cameras.

What's reasonably priced?

You can get the camera itself for less than $20,000. For these guys that's reasonable. In five years I think you'll probably see that ship in a handheld. Several of my larger customers have third party Hasselblad backs that cost about 20 grand, and you have to have a fat pipe to an iMac and a fat pipe to another server. It's not something you can carry around. You can see those at http://www.phaseone.com

Do you have any employees?

I don't have any employees, but I have that independent sales rep; she lives in Massachusetts. And then there's one other person who does sales for the two other products we have: Masterpiece, which is an entry level program, and Picture Perfect which sort of middle of the road. Picture Perfect came from a photographer in Michigan.

But your main sales come from Photo One?

The main dollar volume comes from Photo One. It sells for $1499 for a single, and $1999 for a network, up to 25 users, or $4999 for 26+ users or multiple sites.

Which sells the most?

It's about 60/40 network over single.

Do you sell the hardware too?

No, Stuart used to do that, but I just don't have time for it.

What about training?

We do training, but we handle that a little bit differently. The sales people will go on site and train if they're asked to. Also the day after every trade show we have a full day of free support for our users. We rent a room and a projector and go through the product from A to Z. The customers see a lot of things they've never seen before.

And they'll ask for things and you realize there's something you want to add to the product.

Right, it's a good process for everybody.

Have you ever evaluated Clarion against other development tools?

I've used Delphi a little bit - we've got another program that's written with that. And I've got a fund raising program that's sitting on the back burner. I currently send all the leads that we get on that to Lynn Howard over at Linked Software. His program is written in Clarion. The one we have is written in VB and Access, and it's just not any fun to work on.

What was your first experience with Clarion?

I never had the DOS version of Clarion. My first experience on the PC side of things was with the PC version of COBOL, because I was trying to speed the development of mainframe stuff, which is my background. A buddy at work showed me Clarion for DOS, but I just never had a project where I needed to use it. Then something came up at work, and I got a copy of Clarion for Windows.

What version of Clarion is the current version of Photo One written with?

It's in C5B at present. Mostly AppGen with a lot of embedded hand code.

Did you work with the DOS version of Photo One at all?

I figured I had enough to take on with 700-800 procedures and learning Clarion for Windows. One of the programs was in 1501 at the time, and the other was in 2002.

These were obviously legacy apps. Have you moved to ABC?

It's still a mix. Anything that gets written for both of them is in ABC, and the conversion on both sides is an in-progress thing.

Is the embed code procedural or object-oriented?

It's all procedural. There's a fair number of processes that are built in, like bulk scheduling and a thing we call Recalculate Accounting, which is when someone decides they want to change the way they calculate sales tax, or they want to change from cash to accrual accounting. We have to repost every sale and every cash receipt and every GL entry and that takes a while. I've never seen a piece of code bring a machine to its knees the way that does. The good thing is it goes through the same code that the regular cash postings and GL postings go through, so I don't have to worry about it too much, but it's still kind of an eerie thing.

Are there any third party products in particular you found useful?

We use FileManager 2, CPCS, Arco Word Reporter, RPM, Linder Software compression, and a lot of others. Steve Parker asked how much of this code I actually wrote. When I took over in April of '98 I still had a full time job, so I had somebody do tech support for me. When I came home I'd do the family thing till the kids' bedtime and go start coding. That went on until March of '99, when I left the full time job. I've been here ever since.

Which database driver are you using?

I'm currently using TPS.

You're thinking of making a change?

Not thinking about it – it's a done deal. It's just a matter of time.

Where are you headed?

For our biggest customers, the majority are getting to the point where they have multiple locations so they're using Citrix for that, because TPS across multiple locations is not a good idea. So what we're doing is we're going to move the low-end people to MSDE, and the high end people can go to SQL Server. The next major version will be MSDE and ABC at the same time, so I'll have no hair at the next ETC conference.

Did you consider other options besides Citrix, or was it a foregone conclusion?

We had a customer already using PC Anywhere, and they were having all kinds of problems because they were trying to do production data entry work across a dialup line. It was not pretty. They have seven locations, and I'd had some other experience with the success of Citrix, so they asked me for some options. I said you could do this, or you could do that, or if it was me I'd skip those and go straight to Citrix. So naturally they did the other two things first and didn't like how they turned out. They went with Citrix. They're still on Citrix, at up to seven locations, using the 16 bit version of the app. We've got one other multi-location site that just went live with the 32 bit version in May, and they're really happy with it. The guy who owns the studio went out and bought Citrix stock the day they went live, he was so pleased.

That's quite the recommendation!

He was pretty happy. We've got two other multi-location studios, and they're still trying to figure out how they want to handle those issues. We've got a bunch of code written into the product to handle that, and that's one of the things I need to finish out before the end of the year. There are multiple businesses are in the database and they have multiple sets of preferences, but they can still be reported separately or as a whole.

Designing, writing, and selling software is a complex task. How do you decide where to direct your efforts?

You've got to optimize your time. I spent a long time with mainframe software vendors before I went out on my own, and the one thing I saw consistently among these different companies was that in some cases they tried to do too much themselves. You try to do too much of the marketing yourself, you're designing the logos, and the next thing you know you're spending more time dealing with the infrastructure of your program than you are dealing with the program itself. I have some connections from those years, and I know the artwork they can do, or the marketing they can do, so I don't do much of that stuff. I can do it, those are things that any of us can learn...

But is it the best use of your time?

It's really not. People ask how can you depend on ten or fifteen or twenty third party products? And I say well, I can probably write any one of those, except maybe FileManager 2 just because I don't know what the heck's going on behind the scenes there, but if I was going to do that, I wouldn't have any time for what my business does. I'd spend all my time writing third party stuff.

You've been very successful in relatively short period of time. What's your secret?

I'd say most of it is customer support. These guys were used to being ignored. I'm not here because I'm the world's best ABC coder, much less the world's best legacy coder. Bottom line is I grew up, software-wise, in an environment where service was not only something you needed to do, it was something that was contractually required. I was just baptized into that. I worked for EDS when I got out of school and we had requirements where reports had to be on managers' desks by certain times of day or there were financial penalties. If you got called in the middle of the night with an abend that affected those reports, you were off to work in your skivvies to play with some VSAM files or something. The bottom line is we all get aggravated when we go to a store or any kind of business and the service isn't good. It's not that much harder to do things the right way.

And you obviously get a lot of word of mouth advertising as a result.

Actually the return on investment on our word of mouth is a whole lot better than the return on investment of our advertising. It's a very cliqueish, networky business. There are a lot of speakers in the industry, if you've got the speakers' business and they're saying your name, you can have the biggest, coolest ads in the business and it doesn't matter. You've got to have the names.

That personal recommendation goes a mile.

It sure does. I think that's what got us the Mail Boxes Etc. deal. They're starting a new studio/art gallery franchise. They're calling it Image Arts Etc. They're just getting off the ground now, they've sold about fifteen franchises. They're going to be using Photo One.

That's a slightly different kind of business, isn't it?

Well, kind of. A lot of our customers do framing. A lot of them do tuxes. They'll take your picture, they'll frame it, they'll rent you the tux. The program is laid out pretty well for what Mail Boxes Etc. wants to do with it. As a matter of fact I found about them from Paul Friedman, who I met at ETC this year. He was reading his local paper and emailed me about this new franchise. Well, gee, can't hurt to ask them if they need some software. Next thing you know they're tossing the accounting software they were planning on using and decided to use us.

Networking!

That's what it's about.

If you know of someone you think should be interviewed by Clarion Magazine, send an email to editor@clarionmag.com.

Reborn Free CLARION online published by CoveComm Inc.

Search

Clarion MAGAZINE

The Clarion Advisor: Copying Browse Boxes Between Procedures

Sometimes you need to make a copy of a browse box in another procedure. Unfortunately, Clarion's support for object-oriented development doesn't extend to deriving one browse box from another, but you can accomplish the task with a bit of cutting and pasting.

The key to duplicating browses is in the `#FIELDS` attribute. Go to a browse procedure and click on the […] button beside the Window button. In the window structure look for a `LIST` statement. You'll see something like the following:

```
LIST,AT(8,20,216,124),USE(?Browse:1),IMM,HVSCROLL, ↵
 MSG('Browsing Records'),FORMAT('64R(2)|M~ID~C(0)@n-4@80L(2)↵
 |M~TextField~L(2)@s255@80L(2)|M~BlobField~L(2)@s255@'), ↵
 FROM(Queue:Browse:1),#SEQ(1),#ORIG(?List),↵
 #FIELDS(tes:ID,tes:TextField,tes:BlobField)
```

At the end of the `LIST` is a `#FIELDS` attribute. This attribute defines the link between the listbox's display fields and the fields in the file(s). You can use this information to save a lot of time when copying list boxes.

Jeff Slarve suggests the following steps:

1. Populate a browse template in the procedure where you want to create the copy of the browsebox.
2. Make sure that the correct tables are added to the table tree, along with any local variables that might be needed. This is the thing that's most likely to cause problems, so double check! A good way to verify that it was done correctly is to bring up the listbox formatter to make sure that the fields and local variables are correctly referenced by their respective listbox columns. If not, then ensure that the fields and local variables in question are present in the procedure, then re-paste the `#FIELDS` and re-check in the listbox formatter.
3. Set your range limits, etc.
4. Go to the window [...] of the original browse to be copied.
5. Copy the `FORMAT()` and the `#FIELDS()`

SoftVelocity

FREE Microsoft Internet Explorer

6. Go to the window [...] of the new target browse.
7. Replace the browsebox's `FORMAT()` and `#FIELDS()` attributes with those from the source browsebox.

If you've done everything correctly, the browse fields should all be populated on the new browse.

Main Menu | Log In | Subscribe | Links | FAQ | Advertising

Reborn Free CLARION online published by CoveComm Inc.

Search

Clarion MAGAZINE

Main Page

COL Archive

Log In
Subscribe
Renewals

Frequently Asked Questions

Site Index
Article Index
Author Index
Links To Other Sites

Downloads
Open Source Project
Issues in PDF Format
Free Software

Advertising

Contact Us

The Clarion Advisor: Copying Browse Boxes Between Procedures
(Oct 31,2000)

Give It a Nudge: Adjusting Report Position at Runtime
(Oct 31,2000)

October 2000 News
(Oct 31,2000)

Give It a Nudge: Adjusting Report Position at Runtime

by Lee White and Steve Parker

The technique described here and the code to implement it are entirely the work of Lee White.

Printers are much better than they've ever been.

Printers are much more of a pain than they've ever been.

We get more resolution, more colors, more speed, more options, more features for less money than ever before (at least, until next month). And we get it all in a smaller form factor than ever before.

Why is it, then, that no two printers of the same make and model (possibly even with consecutive serial numbers) can print the same report and look the same? The no-print zone specification, particularly, often seems less a spec and more like a guess, often a bad one, especially when comparing laser to ink jet output.

With standard full-page reports, the movement of bands on the page is often unnoticeable. But, the closer you get to the (theoretical) no-print area, the more likely someone is to report truncation of data. Page headers seem especially prone to losing part of the top line. On labels, this misalignment may render hours or days of painstaking report layout all but useless. The greater the number of labels per page, the more likely printing past the label edges seems to become. And no customer is going to accept "It works perfectly on my printer" (or they'll ask you to send them *your* printer).

The unfortunate fact is that paper handling no longer seems entirely reliable. On some printers, you can actually watch the paper twist as it feeds. But it is our obligation as developers to adjust our code to accommodate.

What's A Developer To Do?

Because each printer seems to be different and users are not always careful about how they load paper, it might seem that there is nothing we can do to control this apparent movement.

But, if you analyze the problem, all that really needs to be done is to allow the user to adjust the relative position of the report. That is, one user may need to nudge the report a bit to the right. Another user may need the report to be a little closer to the top or to the bottom of the page.

Three things are involved here.

First is providing horizontal or vertical displacement of the report.

Second, a way of saving adjustments is needed.

Third, a way of retrieving previously entered adjustments is needed.

In short, what is needed is a little utility to capture user-defined offsets and a way of dynamically applying them. Saving and retrieving can be easily handled by INI files.

The Secret

A report is composed of a series of bands: a page header, a page footer, group headers and footers and details. A report also has a Form.

Each of these has a horizontal and vertical position assigned during formatting. These positions are the first two parameters of the `AT()` attribute when you look at a report structure:

```
HEADER,AT(500,500,7500,500),USE(?PageHeader)
```

or

```
Report REPORT('Customer Report'),AT(500,1000,7500,9000)
```

Actually, details and group headers/footers, for which you will not see an `AT()`, are covered by the `Report` label.

Now, if you need to adjust the Header position to the right by .1", do the following:

```
?Header{Prop:XPos} += 100
```

or, to adjust the header downward by 17 mm:

```
?Header{Prop:YPos} += 17
```

should do the job.

In general, if you have a variable containing the desired movement (the examples are in tenths of an inch, hence the multiplication by 100, adjustments in millimeters can be stored directly and require no further manipulation), you can do this:

```
?Header{Prop:XPos} = ?Header{Prop:XPos} + (var * 100)
```

In mm:

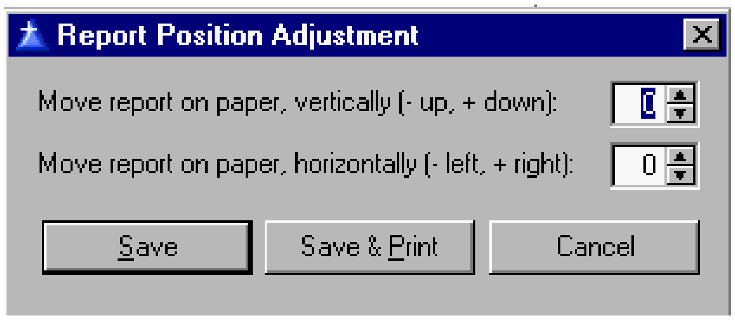
```
?Header{Prop:XPos} = ?Header{Prop:XPos} + var
```

The same transformation would have to be done for the Footer, Form and Report.

Getting User Input

What would be nice is something like:

Figure 1. Capturing user-defined offsets


Report Position Adjustment
Move report on paper, vertically (- up, + down):
0
Move report on paper, horizontally (- left, + right):
0
Save
Save & Print
Cancel

This window is fairly straightforward. However, to make it work correctly, you need to start by loading any previously saved values and end by saving the window values.

Also, to make this more general, you should to save values on a report-by-report basis. This allows "nudges" to be report specific. If one report needs to be nudged into place, adjustments can be stored for that report and applied at runtime. If another report does not require adjustment then nothing need be saved or applied.

This means that a section of the INI file will be labeled by a report identifier.

Figure 2. Retrieving previous entries

```
!LOAD SAVED VALUES
!INI entries are named "report id" + "V"
! or "H" + printer device
!Get vertical offset
INIEntry = 'CustomerReportV-' & PRINTER{PropPrint:Device}
NudgeV = GETINI('ReportNudge',INIEntry,0,'.\nudge.ini')
!Get horizontal offset
INIEntry = 'CustomerReportH-' & PRINTER{PropPrint:Device}
NudgeH = GETINI('ReportNudge',INIEntry,0,'.\nudge.ini')
```

`INIEntry` is a variable that will contain:

1. the report identifier (`CustomerReport`, in this case)
2. "H" for horizontal adjustment or "V" for vertical adjustment
3. the specific printer that needs this adjustment

This makes the adjustment not only report specific but printer specific also and users with multiple printers are covered.

`NudgeV` and `NudgeH` are the vertical and horizontal adjustment with which to start (typically, zero).

When the user completes the window, the values are saved:

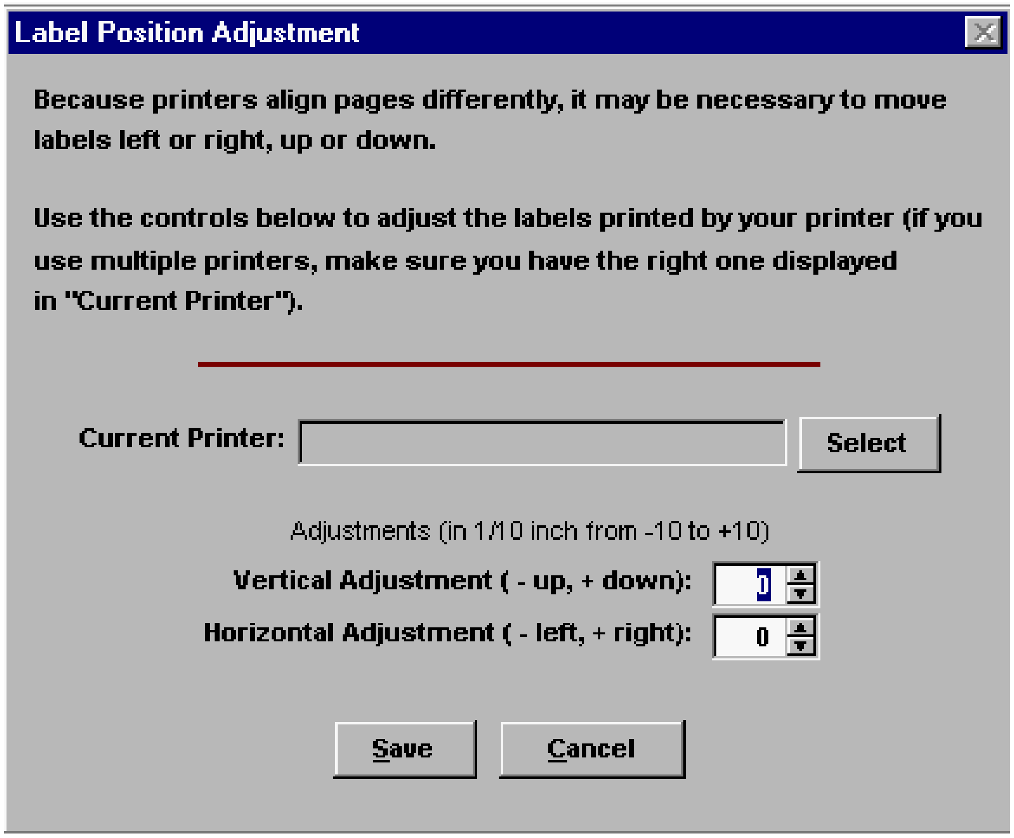
Figure 3: Saving user entries

```
SaveNudge ROUTINE
!INI entries are named "report name" + "V" or
! "H" + printer device name
INIEntry = 'CustomerReportV-' & PRINTER{PropPrint:Device}
PUTINI('ReportNudge',INIEntry,NudgeV,'.\nudge.ini')
INIEntry = 'CustomerReportH-' & PRINTER{PropPrint:Device}
PUTINI('ReportNudge',INIEntry,NudgeH,'.\nudge.ini')
```

The code in Figure 3 takes the report identifier, the printer and the vertical/horizontal value and creates the requisite INI section and entry headers and values.

If you want to let the user choose the printer instead of simply capturing the current printer, the window can be modified to accommodate this need, as shown in Figure 4.

Figure 4: User selects printer



Because printers align pages differently, it may be necessary to move labels left or right, up or down.

Use the controls below to adjust the labels printed by your printer (if you use multiple printers, make sure you have the right one displayed in "Current Printer").

Current Printer:

Adjustments (in 1/10 inch from -10 to +10)

Vertical Adjustment (- up, + down): 0

Horizontal Adjustment (- left, + right): 0

To accommodate user-selected printers, add a local variable and the following code to the button:

```
If PrinterDialog('Select Printer',)
  LOC:Printer = Printer{PropPrint:Device}
  ThisWindow.Reset(1)
End
```

NOTE: `LOC:Printer` should still default to `Printer{PropPrint:Device}`.

This code can be made even more generic by calling the window with a parameter identifying the report. The parameter could then specify the INI file section (that is, the report ID is used for the INI file section) and could be used in formatting the `GetINI` and `PutINI` statements.

Using the report as the section head means that only the vertical/horizontal tag and printer name need be stored in the INI file entry:

```
[PriceTags]
Vert-HP LaserJet III=0
Horz-HP LaserJet III=0
Vert-HP LaserJet 5L PCL=1
Horz-HP LaserJet 5L PCL=0
```

This allows a single procedure to serve any number of reports and printers.

Applying The Nudge

Ok, capturing the user’s needed offsets is accomplished. Now they must be used in an actual report.

Because a Clarion report, like a Clarion window, is actually a data structure, it cannot be affected before it is opened. Therefore, the embed needed is Window Manager | OpenReport, after Parent Call.

First, make sure that the report actually opened and was not cancelled. Then retrieve the required INI file entries, similar to Figure 2, above.

Finally, because a report is never the default target for a property assignment, it is necessary to explicitly set the report as the target for assignments and do the assignments:

Figure 5: Retrieving and setting runtime report positions

```
IF NOT ReturnValue ! make sure report has actually open()ed
!Retrieve INI entries
  INIEntry = 'CustomerReportV-' & PRINTER{PropPrint:Device}
  NudgeV = GETINI('ReportNudge',INIEntry,0,'.\nudge.ini')
  INIEntry = 'CustomerReportH-' & PRINTER{PropPrint:Device}
  NudgeH = GETINI('ReportNudge',INIEntry,0,'.\nudge.ini')
  !Apply to page head
  SETTARGET(Report)
  ?PageHeader{PROP:Xpos} = ?PageHeader{PROP:Xpos} + |
    (NudgeH * 100)
  ?PageHeader{PROP:Ypos} = ?PageHeader{PROP:Ypos} + |
    (NudgeH * 100)
  !Apply to Detail area
  Report{PROP:Xpos} = Report{PROP:Xpos} + (NudgeH * 100)
  Report{PROP:Ypos} = Report{PROP:Ypos} + (NudgeV * 100)
  !Apply to page footer and page form
  ?PageFooter{PROP:Xpos} = ?PageFooter{PROP:Xpos} + |
    (NudgeH * 100)
  ?PageFooter{PROP:Ypos} = ?PageFooter{PROP:Ypos} + |
    (NudgeH * 100)
  ?PageForm{PROP:Xpos} = ?PageForm{PROP:Xpos} + |
    (NudgeH * 100)
  ?PageForm{PROP:Ypos} = ?PageForm{PROP:Ypos} + |
    (NudgeH * 100)
  SETTARGET
END
```

and, voila, the report moves by the indicated amount.

Summary

"WOMM" (Works On My Machine) just doesn’t cut it with end users, even when true. Assuming that the user has the correct printer driver loaded and has good paper stock (in fact some label stock is not perfectly symmetrical) and a report still doesn’t print where it should, runtime adjustment of report positioning is a few embeds away.

The great beauty of this is in the reduction of support calls. The solution to the user’s "problem" is entirely in the app and under the user’s control.

Download the source

Demo app only for C5

Steve Parker started his professional life as a Philosopher but now tries to imitate a Clarion developer. A former SCCA competitor, he has been known to adjust other competitors' right side mirrors - while on the track (but only while accelerating). Steve has been writing on Clarion since 1993.

Reborn Free CLARION online

published by CoveComm Inc.

Search

Clarion MAGAZINE

Clarion News

October 31, 2000

The No DevCon 2000 Subscription Special!
(Oct 31,2000)

The Clarion Advisor: Copying Browse Boxes Between Procedures
(Oct 31,2000)

Give It a Nudge: Adjusting Report Position at Runtime
(Oct 31,2000)

October 2000 News
(Oct 31,2000)

Read The September 2000 News

Clarion 5.5 Goes Gold

SoftVelocity has released Clarion 5.5, and CDs began shipping on October 30th at the rate of 1500-2000 per day. See the new SoftVelocity web site for pricing and support details. XML and ASP add-ons are scheduled for release before the end of the year.

New Clarion 5.5 Newsgroups

There are new Clarion 5.5 newsgroups on the SoftVelocity news server (news.softvelocity.com). Refresh your newsgroup lists!

New SoftVelocity Web Site

The new SoftVelocity web site has been published, coinciding with the release of Clarion 5.5. When fully operational it will contain a knowledge base, news server with web access, and emailing of newsgroup messages to subscribers.

Clarion 5.5 Education Specials

SoftVelocity is offering up to $350 off Clarion 5.5 classes now forming for November and December.

Clarion Handy Tools C5.5 Gold Ready!

The Clarion Handy Tools are C5.5 Gold ready. The current build, (Build "O-4") is available in DLL or source code version for both C5.5 and C5B. Features include: ABC browse enhancements; ABC report and process enhancements; extended window controls (sliders, rulers, address controls, etc); sound and video extensions; IDE control using DDE; application images; Internet tools, API tools, demo applications, help files, and more.

Freeware Word Count Utility

Ralf Schoeffler has released a small freeware utility which will do line and word counts on text files.

New Screen ReDesigner Demo Available

Simon Burrows has released a new demo of his Screen ReDesigner (currently in beta),

DeveloperPLUS
Your source for development tools and add-ons.
Your outlet for application sales.

which lets customers change screen layouts at runtime to suit their own needs. Users can move and resize objects, and change attributes such as font and color. This demo fixes a few bugs and adds a tab sequencing feature.

October 24, 2000

MessageEx Anniversary Offer
MessageEx is now one year old, and is currently on sale for US$39, a 20% discount. This offer is valid until November 23, 2000.

October 17, 2000

Template For Twixtel Phonebook CD
André Gasser is finishing a template to implement direct access to the Twixtel CD's which are distributed in Switzerland. Users will be able to search for addresses in the Twixtel-database directly. The template works with Clarion 5 and 5.5, 32 bit apps, and you need at least version 2.2 Twixtel. Email or phone: +41 (0)56 284 26 26

Buggy 1.2 Available
An update to the Buggy bug tracking tool is available to all registered users. The trial version will also be updated.

New SuperStuff from BoxSoft
Mike Hanson’s new SuperStuff templates are now in their first release. Previously known as the MikeHansonABC template chain, these templates now have better docs and support, and are no longer free (although the legacy version will remain freely available). Updates include numerous changes to Resize support, mhViewManager improvements, better Quicken-style data entry, support for CapeSoft’s MultiProject, and various bug fixes. For all purchase and upgrade issues (including prices and passwords), please contact Mitten Software. Their e-mail address is Mitten@MittenSoftware.com, and their phone numbers are (800) 825-5461 and (612) 745-4941.

Simsoft Templates C4 Compatibility Update
The Simsoft plug-in on-screen keyboard templates are now compatible with Clarion 4. Affected templates include full keyboards, numerical keypad, calculator, and a no-frills calendar. Demo available. Purchase through www.clarionshop.com.

Freeware Snake Game
Chris Jong has released a new freeware implementation of the famous "Snake" game, written in just 460 lines of Clarion code (source included).

ZIPFlash 1.1 Updated With Map Display
ZIPFlash has been updated with several new features, including a map display, which shows the city’s location on a map of the USA, automatic lookup of city/state from ZIP code, updated ZIP data file with latitudes, longitudes, area code and time zone.

Insight Beta 4a Released
Version 1.0 Beta 4a of Insight Graphing is now available to registered users. This update includes a number of fixes, as well as more features, including filters at point level and on queues, a new Pareto graph type (a sideways bar graph), graph sorting, graphing from memory variables, and various bug fixes.

October 10, 2000

Clarion Skill Pool Launched
Have you got a skill you'd like to offer to other Clarion Software Developers? Or maybe you are looking for a person with a particular skill? To help with this process and as a service to the Clarion Community Mike McLoughlin has launched the Skill Pool. To appear in this directory you don't need to be a Clarion Developer - the only requirement is that you can offer services which would be of interest to Clarion Developers. These could be web site design, marketing skills, subcontract programming etc.

New Freeware

Sterling Data has released two new freeware products: zip codes in a TPS file, and latitude/longitudes for 28,000 US cities, also in a TPS file.

SoftVelocity Technical Assistance Plans Adjusted And Discounted

SoftVelocity has adjusted its technical support plan pricing. Sign up for technical support by October 13, 2000 and get an additional ten percent discount. t to kick off the new lineup. We've got the details.

Seeking Clarion Success Stories

Everyone has a great story about how Clarion helped him or her out like no other tool could. SoftVelocity would like to hear yours.

Screen Capture Tools for Clarion

Keystone Computer Resources has just released Screen Capture Tools. These libraries and ABC compliant templates allow the Clarion developer to easily add screen capturing and printing to their applications. Capture screens to the Windows clipboard or save them as BMP files. Print captured images with scaling options, and let the user select an area of the screen to capture. Comes with complete documentation, sample program, and versions for Clarion 5.0 and Clarion 5.5cr2.

CapeSoft TickerTape 1.5 Released

Version 1.5 of the CapeSoft TickerTape control template has been released. This is a free upgrade to all registered users. New features include support for multiple tapes on one window, and support for longer tapes.

New SuperQBE from BoxSoft

SuperQBE 4.90 (ABC) and 2.90 (Legacy) are compatible with C55-cr2. This is the last version to support Clarion 4.0. Includes numerous changes and improvements, including optimized "Begins With" searches, option to show "Be patient" warning message after search is completed, but before filtered browse begins to refill, new embeds, support for CapeSoft's MultiProject, various bug fixes. For all purchase and upgrade issues (including prices and passwords), please contact Mitten Software. Their e-mail address is Mitten@MittenSoftware.com, and their phone numbers are (800) 825-5461 and (612) 745-4941.

New SuperTagging from BoxSoft

SuperTagging 4.90 (ABC) and 2.90 (Legacy) are compatible with C55-cr2. This is the last version to support Clarion 4.0. Features/changes include tag numbers stored as LONGs for greater range, automatic button icons, better file update/delete operations in a process, new embeds, support for CapeSoft's MultiProject, and various bug fixes. For all purchase and upgrade issues (including prices and passwords), please contact Mitten Software. Their e-mail address is Mitten@MittenSoftware.com, and their phone numbers are (800) 825-5461 and (612) 745-4941.

New SuperBrowse from BoxSoft

SuperBrowse 4.90 (ABC) and 2.90 (Legacy) are compatible with C55-cr2. This is the last version to support Clarion 4.0. Features/changes include support for CapeSoft's MultiProject, EIP fixes, and a correction o large format string handling in the ConditionalFormat template. For all purchase and upgrade issues (including prices and passwords), please contact Mitten Software. Their e-mail address is Mitten@MittenSoftware.com, and their phone numbers are (800) 825-5461 and (612) 745-4941.

New SuperInvoice From BoxSoft

SuperInvoice 4.90 (ABC) and 2.90 (Legacy) are compatible with C55-cr2. This is the last version to support Clarion 4.0. Changes include support for CapeSoft's MultiProject and various bug fixes. For all purchase and upgrade issues (including prices and passwords), please contact Mitten Software. Their e-mail address is Mitten@MittenSoftware.com, and their phone numbers are (800) 825-5461 and (612) 745-4941.

New SuperSecurity From BoxSoft

SuperSecurity 4.90 (ABC) and 2.90 (Legacy) are compatible with C55-cr2. This is the last version to support Clarion 4.0. Changes include security support for EIP updates, browse look-ahead to form for security settings, support for CapeSoft's MultiProject, and various bug fixes. For all purchase and upgrade issues (including prices and passwords), please contact Mitten Software. Their e-mail address is Mitten@MittenSoftware.com, and their phone numbers are (800) 825-5461 and (612) 745-4941.

New SuperDialer From BoxSoft

SuperDialer 4.90 (ABC) and 2.90 (Legacy) is compatible with C55-cr2. This is the last version to support Clarion 4.0. Adds support for CapeSoft's MultiProject. For all purchase and upgrade issues (including prices and passwords), please contact Mitten Software. Their e-mail address is Mitten@MittenSoftware.com, and their phone numbers are (800) 825-5461 and (612) 745-4941.

New SuperImportExport From BoxSoft

SuperImportExport 4.90 (ABC) and 2.90 (Legacy) is compatible with C55-cr2. This is the last version to support Clarion 4.0. Changes include better field priming flexibility, drag and drop extension templates, better filter and validation support, improved automatic handling of various data types, and support for CapeSoft's MultiProject. For all purchase and upgrade issues (including prices and passwords), please contact Mitten Software. Their e-mail address is Mitten@MittenSoftware.com, and their phone numbers are (800) 825-5461 and (612) 745-4941.

New SuperFieldFiller From BoxSoft

SuperFieldFiller 4.90 (ABC) and 2.90 (Legacy) is compatible with C55-cr2. This is the last version to support Clarion 4.0. Added support for CapeSoft's MultiProject. For all purchase and upgrade issues (including prices and passwords), please contact Mitten Software. Their e-mail address is Mitten@MittenSoftware.com, and their phone numbers are (800) 825-5461 and (612) 745-4941.

New SuperLimiter From BoxSoft

SuperLimiter 4.90 (ABC) and 2.90 (Legacy) is compatible with C55-cr2. This is the last version to support Clarion 4.0. Changes include support for variable program names, option to turn off the Limiter.Init/Kill calls in the Program Setup/End embeds, improvements to various internal error messages, and support for CapeSoft's MultiProject. For all purchase and upgrade issues (including prices and passwords), please contact Mitten Software. Their e-mail address is Mitten@MittenSoftware.com, and their phone numbers are (800) 825-5461 and (612) 745-4941.

Freeware RAS Dialup Library And Demo App

Ville Vahtera has a new freeware/mailware RAS dialup library available. Functions include RasDial(), to initialize connection; RasActive(), returns true if dial-up active; RasHangUp(), to disconnect current connection; RasEntries(), returns dial-up phonebook entrys; RasList(), returns name of active connection.

ZIPFlash Adds Zip Code Insertion

ZIPFlash is a low-cost way to add ZIP code insertion (with auto-complete) to Clarion applications. ZIPFlash works in auto-complete mode or as a standard lookup, with two lookup buttons on the form. One is for finding the ZIP from the City and State and the other gets the City and State from the ZIP. All source code supplied, and there are no runtime royalties. ZIPFlash does not include the ZIP code database but a copy of this is available in the freeware section of the SterlingData website (ZIPSTPS.EXE). Compatible with all versions from CW2x through C5.5, ABC & Legacy. Cost is $29.