Clarion Magazine

PDF for August, 2002

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

Posted Tuesday, September 03, 2002

How To Ignore A Template: The Browse Reconsidered

In a recent article Steve Parker argued that it was not only possible but desirable (and even wise) to ignore large parts of the Report Template. Challenged by his editor, he now applies this approach to browse procedures.

Posted Wednesday, September 04, 2002

The ClarionMag Limerick Contest

There's been a spate of rhyming in the SoftVelocity chat newsgroup, ranging from doggerel to limericks to haiku. Some range! Favoring the limericking crowd, Richard Rogers has proposed a limerick contest, hosted and judged by Clarion Magazine. (Contest now closed)

Posted Thursday, September 05, 2002

OLE Drag & Drop

If you want to drag a URL from Internet Explorer and drop it onto, say, a Clarion text box, regular drag and drop won't do the trick. You need OLE Drag and Drop. Clarion Magazine's resident master of all things OLE, Jim Kane, shows how it's done.

Posted Friday, September 06, 2002

Weekly PDF For September 1-7, 2002

All ClarionMag articles for September 1-7, 2002 in PDF format.

Posted Monday, September 09, 2002

Data Structures And Algorithms Part VIII - Watching Your Weight

In her previous article, Alison Neal discussed Trees, and more specifically the Binary Search Tree. She also pointed out the main shortcoming of the Binary Search Tree, which is that when data is entered in sorted order, the structure collapses into a linked list. The solution? Weight balanced trees.

Posted Tuesday, September 10, 2002

News

Web-Based Clarion Forum

XP Menubar Template & Classes Release 1

Free XLib-Based Tool

Sterling Data Holiday Schedule

Freeware Calendar

All Icetips Products 20% Off

gFileFind On Sale

Gitano Weekly Drawing Winners

TPS.repair Templates Price Increase

TX Text Control ActiveX v10 Released

Icons & Resources - A free development project

Clarion/PHP Announced

Clarion 5.6 Threading Model Q&A

XP-Like Menu Bar

Demo Clarion Web App

TopSpeed INI Files

INI files are handy, but not always the best way to store initialization information in an already database-enabled application. As it turns out, it's not that difficult to use TPS (or other database files) as drop-in replacements for text INI files in ABC applications. In this article Tim Phillips explains how you can extend the ABC IniClass with just 25 lines of code, and drop in your own INI file handler.

Posted Friday, September 13, 2002

Limerick Contest Results

Here are the results of the Clarion Magazine Limerick contest. Although we received only three entries, all were contenders for the prize.

Posted Friday, September 13, 2002

Weekly PDF For September 8-14, 2002

All ClarionMag articles for September 8-14, 2002 in PDF format.

Posted Wednesday, September 18, 2002

The ClarionMag FAQ - How Many Articles Are There?

Take a look at this photo supplied by Russ Eggen. It shows a double-sided printout of just the 1999 Clarion Magazine articles. A similar printout of the entire site would be a stack of paper over a foot tall!

Posted Wednesday, September 18, 2002

Managing Skipped Field Hot Keys

If you've ever used the SKIP attribute on a seldom-used entry field, you've probably also noticed that you can no longer use a prompt hot key to get to that field. For keyboard-intensive data entry, that's a problem. Fortunately, Carl Barnes has the solution.

Posted Thursday, September 19, 2002

A Tool For Understanding Template Symbols

To be an effective template programmer, you need to know what values various template symbols contain at code generation time. Steffen Rasmussen presents a code template that you insert at any embed point, and which prints out all (or selected) currently available template symbol values.

Posted Thursday, September 19, 2002

A Template For Exporting Classes

Lee White has previously written about creating a template wrapper for classes. That template worked fine, but only in single EXE applications. What about multi-DLL apps? This time around Lee shows how to create a wrapper that handles the tricky business of exporting, and handling

ImageEx Special

Gitano Weekly Drawing

xAppWallpaper Manager v1.6 Released

BoTpl Version 2.2

ComCode Price Increase

Extended Evaluate Released

DynaLib 3.0

Paywire Templates

Skin v1.003b, Documentation, New Examples

Accviz Office

BoTpl 2.1 Update

GeaSoft Clarion Downloads

Freeware Functions Library Updated

Financial Query Scripting Language Update

IAlchemy Free Skins

ProDomus Freeware

Translator Plus Update

BoTpl Free Template Release

Commercial Clarion Apps

TPS Encryption Tips

INN Bio for 5-Sep-2002

Clarion LibMaker 32 Version 2.0 (Freeware)

SealSoft Discounts

New Weekly Drawing

USF By iAlchemy

IconsXP, Theme pack 1, Look Good Package Updates

BoxSoft Super QuickBooks-Export Templates

solid.software Holiday Schedule

Linder Software Vacation Schedule

Skin 1.002 Released

Clarion Freeware

exported, classes.

Posted Friday, September 20, 2002

Weekly PDF For September 15-21, 2002

All ClarionMag articles for September 15-21, 2002 in PDF format.

Posted Monday, September 23, 2002

Data Structures And Algorithms Part IX - Are You Getting Too Tall?

There are two approaches that can be taken to make sure that the Binary Search Tree always provides the best-case scenario. The first is known as a Weight Balanced (or Perfectly Balanced) Tree; the other, and the subject of this edition of Data Structures And Algorithms, is the Height Balanced (or AVL) Tree.

Posted Thursday, September 26, 2002

Advertising Feature: XP Menu - A Simple And Extendable Clarion User Interface

This special advertising feature describes XP Menu, a third party product that brings Windows XP-style menus to Clarion applications.

Posted Thursday, September 26, 2002

How To Ignore A Template

In his recent series on "ignoring" the ABC templates, Steve Parker documented the ABC class methods he uses most. In this article he explains how to discover which ABC methods might be of particular interest to you.

Posted Friday, September 27, 2002

The Clarion Challenge - ContainsMatch

Gordon Smith has proposed a Clarion Challenge, inspired by some code he came across recently. The problem? Write a string matching procedure with some specific requirements. Contest ends Friday, Oct 4, 2002.

Posted Friday, September 27, 2002

Weekly PDF For September 22-28, 2002

All ClarionMag articles for September 22-28, 2002 in PDF format.

Posted Monday, September 30, 2002

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

Anti-Cracking FAQ

Search the news archive

How To Ignore A Template: The Browse Reconsidered

by Steven Parker

Published 2002-09-04

In a recent article, Reports: OOP, ABC and Ignoring Templates, I argued and, I think, demonstrated that it was not only possible but desirable (and even wise) to ignore large parts of the Report Template.

My claim was based on the fact that I consider only four things essential in a template:

- a data section
- a formatter (window and/or report)
- a View (where appropriate)

and

- a processing loop with lots of places for inserting my own code.

All the rest a template provides, *everything* else, is convenience. The bells and whistles of a template constitute the "R" in RAD, affecting the speed of implementation, but the presence or absence of template features (specifically, in the form of template prompts) has no impact on functionality. The templates often make it easier (usually, just plain "easy") to implement business functionality but that is all. They facilitate. The templates do not do anything for you that you cannot do for yourself. And, sometimes, it is easier to do it yourself.

Certainly I like having options available to me on the Procedure Properties worksheet. Yes, I would like the Report and Process templates to offer a "Use standard ‘No Records’ message?" checkbox. Of course I would like these templates to offer me the option of entering my own message (though a custom template doing this isn’t all that difficult; see Template Writing

Made Easier, for example). However, I can see how utterly unwieldy the Procedure Properties could become if a significant number of options were added.

It is equally true that I'm not going to complain if these options are not present. Few take more than a few seconds to implement.

In Reports: OOP, ABC and Ignoring Templates, I described my way of using templates:

> My approach to template-based procedures, I long ago realized, has been to take what the template gives me – or, at least, the parts that work as I expect/need/want them to – ignore the rest to the degree possible and supply the remaining behaviors myself:
>
> … except for the data section, the formatter, the view and the ACCEPT loop, it is entirely possible to ignore the balance of a template generated procedure.

Then my patient, kind-hearted, magnanimous, noble editor (keep them cards and letters coming, Dave, but mostly the cheques – "checks" are also acceptable) asked:

> Would you consider another article applying the same philosophy to, say, a browse? Two things come to mind. One is that you're opening up a lot of people to simply working with ABC objects, outside of the template confines, which is very cool. The other is that it could be a bit trickier to apply this philosophy to a browse or a form, i.e. the templates actually do more that's useful here.

I have never found confession good for the soul but I've a confession to make: "Use what works for you, ignore the rest" is just an intentionally outrageous way of making a point.

Oddly enough, I learned the technique of stating something immoderately in Philosopher School, specifically, in Philosophy of Science.

You all know Newton's "Three Laws of Motion," right? Well, there are really only two. "A body at rest ..." and "A body in motion ..." are the same law. It was explained to me that, in one case, acceleration is zero; in the other, it is not. Consequently the first two laws are not two laws but two cases of one law.

Why, then, did Newton express them as three? Professor Sosinsky explained how this heuristic technique, given the political environment of the day, helped Newton direct people's attention to the points he wanted to make.

Well, my point isn't all that exotic and, certainly, is no secret; actually, it's sort of simple. Stated more conventionally: the ABC templates are little more than wrappers for the ABC

Class Library. That's all, folks.

It is the implication of this that *is* important. The obvious inference is that if the template doesn't do something you want it to do, doesn't have a certain option topside or does something in a way that does not satisfy your needs, it just doesn't matter. All of the ABC Classes needed by that procedure have been instantiated by the templates. Therefore, the properties and methods of those classes are open to manipulation.

C'est tout. And *nowhere* is this more obviously true or more obvious than in a browse procedure.

Ok, mon rédacteur, how's that for "outrageous?"

Browse Basics

A Clarion browse is an extremely complex object. It usually makes use of one or more disk files (or tables), a View constructed from those files and a queue. (You *can* construct a browse using only a queue by populating a list control without the control template and naming your own queue; so, disk files are not a *requirement*).

According the Language Reference Guide, the view is a virtual file, residing in memory. In this respect, it is like a `queue`. Unlike a `queue`, it does not have its own columns but, by `Projection`, has access to file fields and the buffer for the primary file (that is, a view is constructed with the `Project` statement: fields/columns are projected from the underlying file into the view; another way of saying it is that the view is a "file" specification built from the fields and columns of its component files and tables).

The `queue` in a browse procedure is the part you seen on your monitor. Typically, the browse's `queue` is labeled something like `Queue.Browse:1` in template generated procedures and the object name will be something like `BRW1`. The fields in this queue correspond to the underlying View (which was created from the file schematic: primary and any related files).

Thus, the View is the important element. The `queue` is just for show(ing).

Applying the Philosophy

It should be obvious, reading Reports: OOP, ABC and Ignoring Templates, that the vast majority of my manipulations involved sorting and filtering (those which didn't, involved straightforward loop processing). All sorting and filter methods are `ViewManger` methods, affecting the view. Therefore, equally obviously, anything I said about reports involving `AddSortOrder`, `AppendOrder`, `SetFilter` and the like applies equally to a browse.

In fact, my first real use of ABC methods involved browses. A DOS app I was porting to Windows had a file called SpeedSorts. This file contained data constructed from user selections and this data was used to reconstruct the order in which the user wanted records (from another file) sorted.

At runtime, I didn't know whether the user had set up any SpeedSorts or, if they had, how many. I also didn't know how many components a given SpeedSort had (I did know that my predecessors limited users to a maximum of three components). Therefore, it just wasn't possible to use the built-in Conditional Behaviors tab on the Browse Behaviors Property sheet.

Nothing like tackling something simple your first time out of the box, is there?

Using only `SetOrder`, `AddSortOrder` and `AppendOrder`, I found I could dynamically create browse sort orders on the fly from file data, file drops, data collection forms or any other conceivable input device (and showed you how in the series How ABC Handles Multiple Sort Orders, April – June 1999). `SetSort`, a `BrowseManager` method, is the method which changes the order for a browse (usually on a tab change but, now that you know …).

That series of articles also showed how to create tabs for each additional order at runtime and how to ensure that the tabs and sorts merged seamlessly with template-generated tabs and sorts. I mentioned reports in that series, as a four sentence afterthought.

The application handled all of this without knowing, in advance, whether there were any user-selected orders and, if there were, how many. "Dynamic," indeed.

In Order! Order! Order in the Files! (Clarion Online, May, 1999—that seems to have been quite a busy year), I showed how to use `SetOrder` to replace indices (static and dynamic). Only two lines of code are required to sort a browse according to *any* column in the view.

You can implement locators for these *ad hoc* orders courtesy of Jim Kane ("Look Ma, No Keys!," January 2001).

I covered filtering browses, using `SetFilter`, in another three part series, Dynamic Filters (two parts) and SetFilter to the Max (March - April 2001). That series showed how to use `SetFilter` to create filters from the simple to the utterly ridiculous.

All of those articles had sample apps, apps created in 5.0. The information on applying the same philosophy to browses has been there for three years, I won't repeat it here.

Yes, there is one significant difference between reports and browses. You can, for example, perform a calculation in `TakeRecord` in a report. `TakeRecord` is also available in browse

procedures but is less important.

`TakeRecord` is where EOF is checked and a record is read. In a report, it is immediately before the record is printed. So, calculations, lookups, etc., done in `TakeRecord` result in a printed line containing what you expect.

A browse displays a `queue`, a subset of the view. `TakeRecord` won't affect the displayed `queue` record. `SetQueueRecord` does.

End of difference.

Oh, okay, there are some extra embeds that are relevant because forms are sometimes called from browses. But, that's standard language-level stuff; looking at the generated code for `ThisWindow.Run (USHORT Number,BYTE Request)` method makes it clear that you can easily manipulate this method to call alternate update procedures based on a condition or expression (Handling Multiple Update Forms, June, 2001).

I do not understand why it should surprise any one that techniques originally created for browses should transfer to reports (and processes), and *vice versa*. I do not understand at all: browses, reports and processes all instantiate a `WindowManager` and a `FileManager`. The methods of these classes don't change just because one procedure contains a displayable `queue` and another contains a `Print` statement and a third contains neither...

In fact, a Source procedure may be the only one not derived from a Window.

In Sum

So, to finally answer the question that was asked, "no, I won't consider another article." I don't have to, it's already been written.

I will, however, consider restating what I want to communicate: the ABC templates are just wrappers for the ABC Classes. Therefore, it's the methods … fella.

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

Reader Comments

Add a comment

You all know Newton’s "Three Laws of Motion," right?...

The ClarionMag Limerick Contest

Published 2002-09-05

This contest is now closed. See who won!

There's been a spate of rhyming in the SoftVelocity chat newsgroup, ranging from doggerel to limericks to haiku. Some range! Favoring the limericking crowd, Richard Rogers is throwing down the following challenge, hosted and judged by Clarion Magazine.

> Here's an offer to you gals and guys.
> The Sylkie will offer a prize
> to the best loop or case
> with a limerick base
> and we'll hope that there aren't any ties!

The winning entry will receive either a license for App Init, or a cool little notes database, from Richard. The rules are as follows:

- The limerick itself must be compilable Clarion source code, in the form of an IF, LOOP, or CASE structure.
- Words to be used can be declared (apart from the verse) as equates, data labels, calls to external routines, etc.
- Points given for actual usefulness of the code, rhyme, meter, double-entendre (but please remember this is a family magazine), etc.
- Line continuation characters are considered punctuation.
- You may only use a comment on the last line.

An Example (you can do better than this, right?):

```
  DATA
IntoTheBrew  LONG(0)
Land         LONG(0)
```

```
Sea          EQUATE(100)
Water       EQUATE(5)
Tea           EQUATE(5)
Sugar            EQUATE(1)
  CODE
Loop until Land = Sea
   Land = Water + Tea
   IntoTheBrew = |
   Sugar + 2
END ! Of Loop and an exit for me
```

The contest ends midnight GMT, September 8, 2002. Send your entries to limerick@clarionmag.com.

This contest is now closed. See who won!

Reader Comments

Add a comment

OLE Drag & Drop

by Jim Kane

Published 2002-09-06

A while ago someone on the newsgroups asked how to drag a URL from Internet Explorer into a Clarion text box. The technology needed to do that is OLE Drag and Drop. More specifically, the Clarion text box needs to implement an `IDropTarget` OLE interface. (If you have not used interfaces much yet, you should read Phil Will's recent article. It is very clear and well done.)

The only difference between implementing an ordinary interface and a OLE interface is that in the case of the OLE Interface, you have to add the "`COM`" modifier to the interface definition and implement the standard `IUnknown` methods (`QueryInterface`, `AddRef`, and `Release`):

```
IDropTarget         INTERFACE,COM,type
QueryInterface        PROCEDURE (long  iid_Requested, |
                         *LONG lpInterface),long
AddRef                PROCEDURE (),Long,PROC
Release               PROCEDURE (),Long,PROC
DragEnter             PROCEDURE(long lpIDataSource, ulong grfKeyState, |
                         long lpPtx, long pty, *long DWEffect),long
DragOver              PROCEDURE(ulong grfKeyState, long Ptx,long Pty, |
                         *long DWEffect),long
DragLeave             PROCEDURE(),long
Drop                  PROCEDURE(long lpIDataSource, ulong grfKeyState, |
                         long Ptx,long Pty, *long dwEffect),long
                    END
```

And the class that implements the interface is defined like this:

```
DropTClType Class(StdComCltype),type,module('DropTCl.CLW'),|
  LINK('DropTCl.CLW',_ABCLinkMode_),DLL(_ABCDllMode_),|
  implements(IDropTarget)
```

To create the DropTClType drop target class you have to write a method for every interface method, whether or not the method is actually used. Fortunately, that is quite easy. The sample code in the accompanying download shows how. The DropTCl class that implements the IDropTarget interface derives from StdComCl. I've written about that class in many other articles; its job is to provide some basic COM support and error handling.

The QueryInterface method's job is to tell the world that this an IDropTarget interface, should anyone ask. AddRef and Release simply maintain a reference count that isn't used. Regardless of the fact the methods are not important, they must be present and implemented, since all COM object are expected to implement them.

Implementing a COM interface takes a little getting use to. Normally when you create a class, you call it. In this case, you will never call any of these interface methods. Instead, Windows will call them when a user has dragged something over your drop target. In order for Windows to know about your IDropTarget interface, you have to register it. That is taken care of in the DropTCl.Init() method, which calls OLEIntialize to tell Windows this thread is going to use OLE. For drag and drop, you must call OLEInitialize and not Cointialize to initialize COM. While a simple matter, my not knowing that delayed this article by about three years! Next you must call RegisterDragDrop(hwnd,address(YourClass.IDropTarget)), passing both the hwnd for the control to act as a drop target, and the address of your implementation of IDropTarget. Any Clarion control for which prop:handle returns a hwnd should work. The Kill method does the reverse of the Init method and cleans everything up.

Understanding drag and drop

A quick walk through of the drag methods will give you an idea of how drag and drop works. DragEnter is called by the RTL when a user drags something over the control serving as a drop target. When Windows calls your IDropTarget.DragEnter method, you get a chance to find out what kind of data is hovering over your control. If the data isn't something you want, you just set the DWEffect parameter DragEffect_None and return. That cancels the drop. The cursor will change to the typical no drop icon.

If you can make use of the data available, then set DWEffect to DragEffect_copy, DragEffect_Move, or DragEffect_Link and return. You can make use of grfKeyState parameter to DragEnter to determine what keys are depressed. For example, if the control key is pressed Band(grfKeyState,MK_Control) will be true. The equate values for the various MK_keyvalues and DragEffect values are in the source code of the sample application. Of course, you are under no obligation to pay any attention to grfKeyState if you do not want to. The entire purpose of DragEnter is to see what data is available and to allow you to tell Windows if you are willing to accept any of

the data by setting `dweffect` to one of the values mentioned, or to `DragEffect_none` to cancel the drop. Below is a further discussion of how Windows tells you about the data available and its format.

If `DragEnter` does not return `DragEffect_None`, then the drag is still on and window calls `DragOver` periodically while the cursor is over the drop target. If necessary you can check the mouse position passed in the `ptx` and `pty` paramters and, if desired, scroll a list box or other scrollable control if the cursor is near the edge. Because this method is called periodically, you can think of it as similar to a Clarion timer event.

Controls located on, or covered by, a Clarion sheet control provide a special challenge. You must make the sheet the drop target, and then in `DragOver`, determine if the tab the control on it is active, and if the cursor is over the control within the sheet that is the true drop target. The initialization code in the sample application to set this up looks like this:

```
accept
  case event()
  of event:openwindow
    !allow drop on the sheet
    Droptcl.init(?sheet1)
    !restrict to the text control on the sheet
    DropTcl.limitDropArea(?text1)
    !only allow a drop when tab1 is active
    DropTCl.RestrictToTab(?Sheet1, ?Tab1)
```

In `event:openwindow`, the sheet is set to be the drop target, and then the drop target area is limited to the area covered by a text control (`?text1`). Further, the drop only works when `?tab1` is chosen on `?sheet1`. If you needed to allow drag and drop to more than one control on the sheet, you would need to modify the code in `LimitDropArea` to allow for an array of controls. You would then have to check each control in turn to see if it was under the cursor.

Since `RegisterDragDrop` associates a particular control or `hwnd` with a particular `IDropTarget` interface instance, the same `hwnd` or sheet cannot be associated with more than one `IDropTarget` interface, otherwise when the cursor is over the sheet, Windows won't know which `IDropTarget` interface to call. Because of that Windows-imposed limit of one `IDropTarget` interface per `hwnd`, you could not create a second instance of the `DropTCl` to handle a second control on the same sheet. If you tried, the `DropTCl.Init` method would fail because `RegisterDragDrop` would fail when trying to associate a second `IDropTarget` interface with the same hwnd.

The code in `DragOver` looks like this:

```
DropTClType.IDropTarget.DragOver  Procedure(ulong grfKeyState, |
long Pointx,long Pointy, *long DWEffect)
```

```
point &pointtype
  code
  !See if over the target
  if SELF.CheckLimitArea(pointx, pointy) then
    return e_unexpected
  end
  {some code omitted}
  return 0
```

Pointx and Pointy store the cursor position relative to the upper left of the screen (not the upper left of the sheet, or anything else). Returning e_unexpected tells the caller the drop will not happen so it changes the cursor back to the no drop shape. A return value of 0 from any COM method indicates success, and is usually written as S_OK. To determine if the cursor if over the control of interest, convert the screen coordinates to coordinates relative to the text control:

```
DropTClType.CheckLimitArea procedure(long ptx, long pty)
POINT GROUP
PTX LONG
PTY LONG
  END
!return values
! 1= do not allow the drop
! 0= allow the drop
  code
  {some code omitted for clarity}
  point.x=ptx
  point.y=pty
  if ~ScreenToClient(SELF.hwndlimit,Address(point)) then return 1.
  if ~getclientrect(SELF.hwndlimit,address(rect)) then return 1.
  if ptinrect(address(rect),point.x, point.y) then
    return 0
  end
  return 1
```

The coordinate conversion is done with the screenToClient API call. SELF.hwndlimit is the hwnd of the text control. GetClientRect gets the size of the text control where the left and top members of the rectangle are 0 and the right and bottom members are the width and height. I could have used equivalent Clarion code, but GetClientRect was convenient. PtinRect determines if the cursor position converted to be relative to the text control is within the area of the text box. For example, if the converted cursor coordinates are negative, then the cursor is above or to the left of the text box.

Certainly this code could be expanded to provide finer control. For example, you could limit drops to a single column in a listbox. Likewise you could use knowledge of row height to determine what row you were over. If the cursor was over the top or bottom 100 pixels of the control and the control was scrollable, you could scroll the control.

If the cursor leaves the area over the drop target without releasing the mouse, `DragLeave` is called. Again, there is very little code in `DragLeave`, and that typically is just for clean up.

`Drop` is called when the user releases the mouse over the drop target. When `drop` is called you read the actual data from another interface called `IDataObject`. The only cautionary note is Clarion does not do well if you try to store the data in a threaded file or if a timer is running elsewhere in your application (which forces periodic thread switching). Depending what you are trying to accomplish, very often all you do with the data is add it to a queue or display it in the control where the drop happens, and that should work just fine. Perhaps Clarion 5.6 will solve some of these issues with timers and threaded files; time will tell.

If you have been looking at the sample code, perhaps you have noticed one other oddity. I haven't declared groups as local data; instead, I've used this kind of declaration:

```
!group definition
FORMATETCtype       group,type
cfFormat            ushort
dummy               ushort
DVTARGETDEVICE      long  !pointer
dwAspect            ulong !1=content,2=thumbnail,4=icon,8=docprint
lindex                  long  !-1=all
tymed                  long
    end
```

I then call the `MakeGroup` method when I need a group:

```
!create the groups
  formatetc&=(SELF.MakeGroup(size(FormatEtcType)))
```

Here's the code for the `MakeGroup` method, which uses `new()` to create the group:

```
DropTclType.MakeGroup procedure(long pSize)
tempstr &string
  code
  tempstr &= new string(pSize)
  assert(~tempstr&=NULL)
  return (address(tempstr))
```

`MakeGroup` returns the address of the `GROUP`'s location in memory; the calling code then cast the address to a reference variable using the familiar `Reference &=( some address)` format. Casting groups is not fully supported in Clarion. You can access fields in the resulting reference variable, but code like `clear(groupvariable)` will not work. To compensate for that, I added two new methods - `ClearFormatETC` and `ClearStgMedium` - to clear the groups field by field. If using casting which isn't fully supported bothers you, you could, with just a little more overhead, create a class that contains nothing but a group and new the class. You would then need to use `class.group` syntax

through out the code.

The reason for all this is that the OLE methods are sensitive to the memory address the groups are located on. By default Microsoft compilers allocate a group of longs on a four byte boundary (i.e. `address(group) % 4=0`). These OLE functions require groups be on four byte boundaries, at least on some operating systems, most notably Windows NT. To make the code work universally, I use `New()` since it always allocates memory on eight byte boundaries.

Before I realized `NEW()` used 8 byte boundaries, I declared my groups statically. Then I'd make the tiniest change to the local data and it wouldn't work. I couldn't be sure if it was a Clarion bug where code would fail when data was aligned on certain boundaries or a Windows problem, so I enlisted Alexey Solovjev's help. He was kind enough to first confirm I wasn't crazy (on at least this issue) and he tracked it down to a particular and rather obscure API call: `NdrComplexStructBufferSize`. Alexey recommended using the class wrapper since casting groups isn't fully supported, but to minimize impact on my code I went the casting route. In any case, using `new()` was a convenient way to ensure data was aligned on a boundary Windows could live with. The scary part of it is there was no special documentation or mention of this requirement, so if you ever see code failing when you are convinced it shouldn't be and there are groups involved, consider data alignment as a possible cause. This seems to be especially true for the network management APIs (which start with the letters 'Net') and RPC APIs. Further there is a MSDN article called 'Storage and Alignment of Structures' that explains alignment requirements for groups pretty well. Yet another lesson learned the hard way.

The most difficult thing about drag and drop code is that the dropped data can be presented in various formats, and you don't know at compile time what those formats will be. When `DragEnter` is called, you are passed a pointer to a `IDataObject` interface.

The code in the sample application casts the pointer (`lpDataObject`) passed to you as a parameter to `DragEnter` into the `IDataObject` interface. It then calls the `IDataObject.EnumFormatEtc` method to get yet another interface called `IEnumFormatETC`. `IEnumFormatETC` is a lot like a simple version of a Clarion queue, and by calling its `Next` method you can work through all the format information it contains in a loop. The code looks like this: (error checking removed):

```
Idataobject&=(lpIDataobject)  !cast pointer to interface
   !get IEnumFormatETC
    IDataObject.EnumFormatEtc(1, lpIEnumFormatetc)
    IEnum&=(lpIEnumFormatetc)
    !loop through IFormatETC looking at all the available data
    loop
      clear(formatetc)
```

```
      lpFormatEtc=address(formatETC)
      IEnum.Next(1,lpFormatetc,count)
      if count<>1 then break.
     !---process FormatEtc here---
  end
```

The `FormatETC` group returned by the `IEnumFormatETC.Next()` method describes the data available by providing its clipboard format, and telling you how it is stored. `FormatETC` contains a `cfFormat` field which represents the clipboard format as an integer. To convert the clipboard format provided in the `FormatEtc` group to a readable format call the `GetClipboardFormatName()` API:

```
!Get the  data's clipboardformat
if ~getclipboardFormatName(formatEtc.cfFormat,|
   cFormatname, size(cformatname)) then
  clear(cFormatName)
end
!cFormatName contains the readable format of the clipboard format
```

The `cfFormat` values should not be stored. Aside from a few predefined ones, they are created when the drop source registerers a certain clipboard format, and will frequently vary each time your program is run.

All the above code that gets the `formatETC` structure and the clipboard format name is boilerplate code and unlikely to change. The important thing is what you do with this information when you get it. All the available information about the type of drop data available is passed to a virtual method for a decision. In you main application you will have code like this to add the `IDropTarget` interface to a text control.

```
dropTcl         class(DropTClType)
onDataDrop        procedure(long lpData),byte,derived
OnFormatDetected  procedure(*cstring cFormatname, |
                     long cfFormat,long Tymed,|
                     long grfKeyState, *long dwEffect)|
                     ,byte,derived
                  end


Text1 cstring(2000)

window WINDOW('Drop Test'),AT(,,260,100),GRAY,DOUBLE,System
         TEXT,AT(20,8,199,48),USE(Text1)
       END

  code
  open(window)
  display()
  accept
    case event()
    of event:openwindow
      Droptcl.init(?sheet1)
```

```
      DropTcl.limitDropArea(?text1)
      DropTCl.RestrictToTab(?Sheet1, ?Tab1)
    end
  end
  droptcl.Kill()
  close(window)
```

`OnFormatDetected` is the virtual method called from `DragEnter` when all the data about one of the available drop formats is available. The format name is the best thing to use for identifying the data type. There are a few standard `cfFormat` values like `cfFormat=1`, which means the format is plain text. For the most part, however, `cfFormat` values are variable. `Tymed` describes how the data is formatted. Some common `Tymed` values are:

```
!    TYMED_HGLOBAL      = 1,
!    TYMED_ISTREAM              = 4,
!    TYMED_ISTORAGE     = 8,
```

Of these `hglobal` is the easiest to deal with, as you will see. Streams and storages are a bit more complex and require mastering the `IStream` and `IStorage` interfaces to use. The sample code does not use them. While there are other possible values, these are the only ones I've ever seen used.

After a bit of experimentation you will find URLs are typically dropped with either a clipboard format of '`UniformResourceLocator`' or plain text (`cfFormat=1`, a predefined Windows value) and have a `tymed` of 1. To respond to these kinds of drops the code is as follows:

```
DropTCl.OnFormatDetected procedure(*cstring cFormatname, |
   long cfFormat, long Tymed, long grfKeyState, |
   *long dwEffect) !,byte,virtual
res byte(5) !returning this value causes the enumeration to continue
  code
      !for debugging and exploring only:
      !junk#=parent.OnFormatDetected(cFormatName, |
       !cfFormat, tymed, grfkeystate, dweffect)
      !Look for a uniformresourcelocator
      if upper(cFormatName)='UNIFORMRESOURCELOCATOR' |
           and band(Tymed,1) then
        !For URLs we need to set link
        res=0  !returning this causes this value
               !to be approved for drop
        SELF.cfFormat=cfFormat
        dwEffect=bor(dwEffect,DropEffect_Link)
      elsif cfFormat=1 and band(Tymed,1) then
        res=0  !returning this causes this value
               !to be approved for drop
        SELF.cfFormat=cfFormat
        dweffect=band(dweffect,0FFFFFFF0H)
        dwEffect=bor(dwEffect,DropEffect_copy)
      end
```

```
      return res
```

When `OnFormatDetected` receives an appropriate drop format, it prepares a return value of 0 and sets `dwEffect` to `DropEffect_Link` or `DropEffect_Copy`. For reasons not immediately obvious the drop of a data type of '`UniformResourceLocator`' will only work when `dwEffect` is set to `DropEffect_Link`. Plain text drops worked with any of the drop effects. To test a plain text drop, open wordpad (not notepad), type some text, highlight it, and drag it over the text control in the sample app.

The only way to find out about what drop formats are available from a certain program is to experiment. If you uncomment the call to parent.`FormatDetected()`, all the various drop formats available will be displayed in a message box one after the other. Unfortunately the message boxes ruin the drop, but that is just fine for experimentation.

After you approve a particular drop format by setting `dweffect` and returning zero from `OnFormatDetected`, and as soon as the user releases the mouse, the `Drop` method will be called. Once again a pointer to `IDataObject` is passed to you by Windows in the `Drop` method call. First a `FormatETC` group is set up, using the clipboard format you picked in `DragEnter` plus the tymed or data storage type you want. Please consult MSDN for all the details on the other fields. For most purposes the standard values used in the sample code will suffice. `QueryGetData` is first called to see if the data format and storage format you want is available. While it usually is, the conditions at the drag source could change at any time, and it is possible the data is no longer available. If it is available `GetData` is called and a stgmedium group is returned. This group contains a handle to the global memory block (an `hglobal`) containing the dropped data. It just takes just one API call to lock the global memory block and get a pointer to the data. The pointer is then passed to the `OnDataDrop` virtual method

```
lpData=(globallock(stgmedium.hglobal))
if ~lpdata then res=return:fatal else
  res=SELF.OnDataDrop(lpData)
end
releaseStgMedium(address(StgMedium))
```

The call to `releaseStgMedium` frees the memory containing the dropped information.

Back in your application, in the `OnDataDrop` method you can do what ever you want with the data. As you can see, in the sample code, the data is copied to a cstring which is displayed in a text control using the `lStrCpyN` API call. `lStrCpyN` copies a maximum of `size(Text1)` bytes from `lpData` into `Text1`. This eliminates the possibility of a buffer over run, which is important since you have no control over the size of the data dropped:

```
droptcl.OnDataDrop procedure(long lpdata)
  code
```

```
  lstrcpyn(address(text1),lpdata, size(text1))
  display(?text1)
  return 0
```

To test, compile and run the sample application. The application has one window with a text box on it. Next open Internet Explorer. Put your mouse over a link and depress the left button and begin to drag. You will see the cursor change to the typical drag and drop shapes. When the cursor gets over the text box, the cursor will change to the cursor indicating a link. If you release the mouse while over the text box, you will see the url. Likewise the sample is also coded to allow drops of plain text. Open WordPad (not notepad – it's not an OLE drag source), highlight some text and drag it to the text control, and release to replace the text in the control with the dropped text.

While certainly this is a bare bones demo, there is quite a lot you can do with this code. As you may have noticed, in the `Init` method an `IDropTargetHelper` interface was created. While this isn't supported on all operating systems or by all drag sources, when available it provides an alpha blended or ghosted image of the item being dragged (this is the kind of thing you see when dragging files and folders around in Windows explorer). It is very little work to implement. Every time Windows calls one of your `IDropTarget` methods, just pass the same parameters on to the `IDropTargetHelper` interface if it is available.

Also notice that most of the methods of `IDropTarget` provide you with mouse coordinates in the `ptX` and `ptY` parameters. You can convert these to X and Y positions relative to the upper right corner of you control using the `ScreenToClient()` api. Once you know the mouse position, you can calculate the line of a list box the drop is over. You can then choose to scroll the listbox if the cursor is near the edge or perhaps change the `dwEffect` value depending on exactly what the drop is over within your control. If you're dropping over a text box, you could probably calculate the location within the text you are over and insert the text at that spot rather than replacing all the text as this demo does. So depending on your needs you can add just about any feature you've seen in any other Windows program without much additional work. Not bad for surprisingly little code. I just wish it didn't take me three years to get the `Init` method to work, not to mention the data alignment issues! Details, details.

Download the source

Jim Kane was not born any where near a log cabin. In fact he was born in New York City. After attending college at New York University, he went on to dental school at Harvard University. Troubled by vast numbers of unpaid bills, he accepted a U.S. Air Force Scholarship for dental school, and after graduating served in the US Air Force. He is now retired from the Air Force and writing software for ProDoc Inc., developer of legal document automation systems. In his spare time, he runs a computer consulting service, Productive Software Solutions. He is married to the former Jane Callahan of Cando, North Dakota. Jim and

Jane have two children, Thomas and Amy.

Reader Comments

Add a comment

Seems like it would be good to ASSERT that the RTL is...

Topics

Data Structures And Algorithms Part VIII - Watching Your Weight

by Alison Neal

Published 2002-09-10

In my last article I discussed Trees, and more specifically the Binary Search Tree. I also pointed out the main shortcoming of the Binary Search Tree, which is that when data is entered in sorted order the structure collapses into a linked list. This failing defeats the purpose of the Binary Search Tree, which is to provide the ability to find data elements quickly. When you want to find an element in the List structure, for instance, you have to start at the head every time and visit every single node until you reach the place where the data element you're trying to find should be logically located. The best-case scenario for a Binary Search Tree should effectively halve the number of nodes you have to visit.

There are two approaches that can be taken to make sure that the Binary Search Tree always provides the best-case scenario. The first is known as a Weight Balanced (or Perfectly Balanced) Tree, and the second is known as a Height Balanced (or AVL) Tree.

Weight Balanced Binary Search Tree

What is a Weight Balanced Binary Search Tree? Figure 1 shows an example of a tree that is *not* weight balanced.

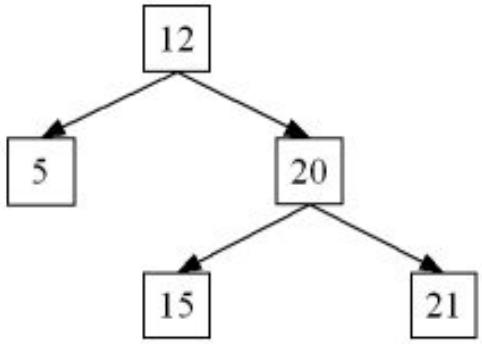


Figure 1. A non-weight balanced binary search tree

In Figure 1 the weight of the left hand side is one and the weight of the right hand side is three; the weight measurement is derived from the number of nodes on either side of the Tree.

A Binary Tree is considered "perfectly balanced" if for every node in the tree the number of nodes in its sub trees differs by at most one. This means that the weight of the sub trees should only differ by at most one. Here are some examples of Perfectly Balanced or Weight Balanced Trees.

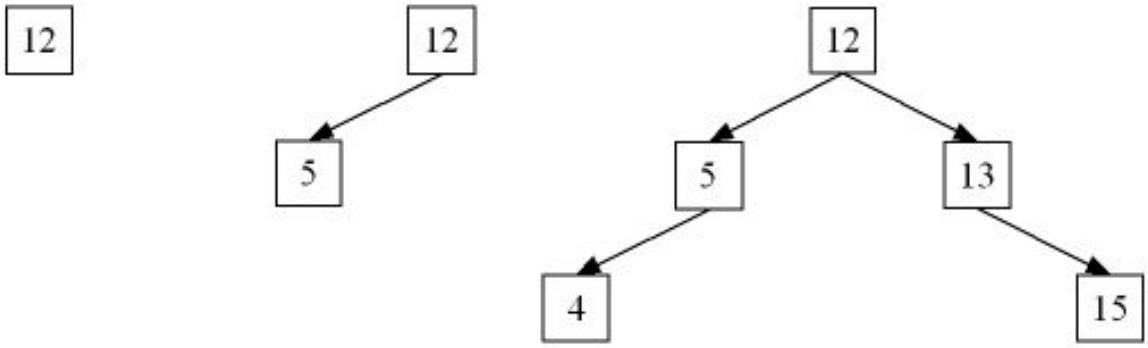


Figure 2

In the first tree in Figure 2 there is only one node, so it is perfectly balanced. In the second tree there are two nodes; the left hand side has a weight of one and the right hand side has a weight of zero. This is balanced because the weight difference is not greater than one. The third tree has five nodes and both sides have a weight of two.

Unfortunately, adding and removing nodes from a Weight Balanced Binary Search Tree is not easy, because you not only have to keep the data in order, you must also keep the weight balanced. This should be a hint that the Weight Balanced Tree may not be a good choice for frequent additions and deletions.

The methods I’ve used in my implementation of the Weight Balanced Tree are as follows:

Method	Description
`Init`	Initialise the root of the tree
`Kill`	Dispose of the tree
`IsEmpty`	Check to see whether the tree is empty
`Insert`	Add a data item to the tree
`Height`	Get the height of the tree
`Find`	Find an element in the tree

The `init`, `Kill`, `isEmpty`, `Height` and `Find` methods are exactly the same for the Weight Balanced Tree as they are for the normal Binary Search Tree covered in my last article.

The `Insert` method, however, is a different story. I've assumed that I will only be loading the tree once and that I know the total number of nodes required from the outset. There are two methods you can use with in this instance; the first is to simply sort the data prior to inserting it into the tree. To do that you would simply recurse to the left most node insert the first element (assuming lowest value first order), then recurse right. The code would look something like this:

```
PBInsert                (MyValQ ValQ, *treenode t, Ulong SZ, *ULONG i)
NLeft           ULONG
NRight  ULONG
  CODE
  IF SZ = 0 THEN RETURN NULL.
  NLeft = SZ - 1 /2
  Nright = SZ - Nleft - 1
  IF t &= NULL
    SELF.Curr &= NEW(treeNode)
?   ASSERT(~SELF &= NULL)
    t &= SELF.Curr
    t.lNode &= NULL
    t.rNode &= NULL
  END
  t.lNode &= SELF.PBInsert(ValQ,t.lnode,nLeft,I)
  I+=1
  GET(ValQ,I)
  t.nodeVal = ValQ.thisVal
  t.rNode &= SELF.PBInsert(ValQ.t.rnode,nRight,I)
  RETURN t
```

Otherwise you can maintain the data in order, as the following code shows:

```
Pinsert  PROCEDURE(ULONG yourVal,ULONG SZ)
done    BOOL
  CODE
  SELF.root &= SELF.PBinsert(SELF.root,yourVal,SZ,done)

PBinsert PROCEDURE(*treeNode t,*ULONG yourVal,ULONG SZ,*BOOL done)
nLeft   ULONG
nRight  ULONG
  CODE
  IF SZ = 0 THEN RETURN NULL.

  nLeft = (SZ - 1) / 2
  nRight = SZ - nLeft - 1
  IF t &= NULL
    SELF.curr &= NEW(treeNode)
?   ASSERT(~SELF.curr &= NULL)
    t &= SELF.curr
    t.ltree &= NULL
    t.rtree &= NULL
    CLEAR(t.nodeVal)

  END
  IF ~done AND nLeft
    t.ltree &= SELF.PBInsert(t.ltree,yourVal,nLeft,done)
  END
  IF ~done
    IF ~t.nodeVal
      t.nodeVal = yourVal
      done = TRUE
    ELSIF t.nodeVal > YourVal
      SELF.swap(t.nodeVal,yourVal)
    END
  END
  IF ~done AND nRight
    t.rtree &= SELF.PBInsert(t.rtree,yourVal,nRight,done)
  END
  RETURN t
```

Taking the following set of numbers:

48	10	6	14	4	46	21

The first call to `Pbinsert` passes the value 48, and the required number of nodes, 7. `Pbinsert` then makes a recursive call to `Pinsert` (Step 1). `SZ` is 7 so `nLeft` is calculated to be 3 and `nRight` is calculated to 3. This provides the weight requirements for each side of the tree. The first `Node` is allocated but no value is assigned yet. As `nLeft` is 3 and the value 48 still hasn't found it's resting place (not done) then a recursive call to the left of the tree is made. This time (Step 2) the passed `SZ` value is 3 so `nLeft` is calculated to be 1 and `nRight` is calculated to be 2. Another node is then allocated with no stored value. Another recursive call is made to the left because `nLeft` is 1 (Step 3). This time `nLeft` and `nRight` are both 0. A new node is allocated but because `nLeft` is zero there is no recursive call made.

The next `IF` statement executes because 48 has still not been allocated. So does the nested `IF` statement, because the current node does not have a value allocated to it; 48 is allocated to the left most node (the position of the lowest number in the tree).

Step	r	Size	YourVal	nLeft	nRight
1		7	48	3	3
2	1	3	48	1	1
3	2	1	48	0	0

My Tree now looks like Figure 3.

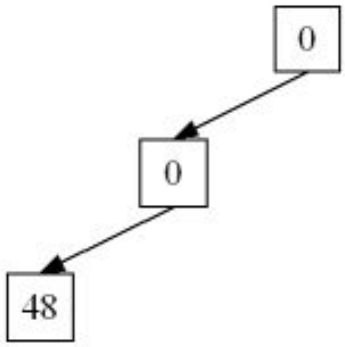


Figure 3

As everything is now done there are no more recursive calls, and the functions return successfully.

I have stopped a large number of recursive calls from occurring by checking to see whether (1) there is any point in recursing (whether the `nLeft` or `nRight` values are greater than 0) and (2) the process is "done" yet or not. You don't have to perform these comparisons if you don't want to. If you don't, the recursive calls will stop on their own when `SZ` reaches zero, and all the required nodes will be allocated when the first number is added to the tree. However, my approach may introduce more comparisons than really required, and thus slowing the process down.

When the value 10 is added, `pbInsert` is called with `SZ` being 7 (Step 4).

Step	R	Size	YourVal	nLeft	NRight
4		7	10	3	3
5	1	3	10	1	1
6	2	1	10	0	0

Again, the recursive calls head to the left most node, and then compare the previous lowest number of 48 with the new value 10. As 10 is less than 48 it is swapped. However, the third call returns (going back to step 5) and 48 hasn't found its resting place in relation to 10 yet, so the value is stored in the second to left most node, and the process is considered done (Figure 4).

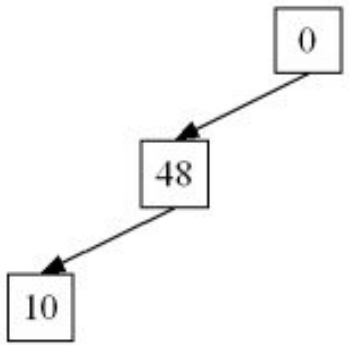


Figure 4

The next number to add is the value 6. Again recursive calls are made to the left, and 10 is compared with 6 and swapped (Step 9).

Step	R	Size	YourVal	nLeft	NRight
7		7	6	3	3
8	1	3	6	1	1
9	2	1	6	0	0
10	3	1	48	0	0

The function returns up one level and the value 10 is then compared with 48 and swapped. As the process is not yet considered done and `nRight` is 1, another recursive call is now made to the right, the new node is allocated, and 48 is assigned. Now the Tree looks like Figure 5.

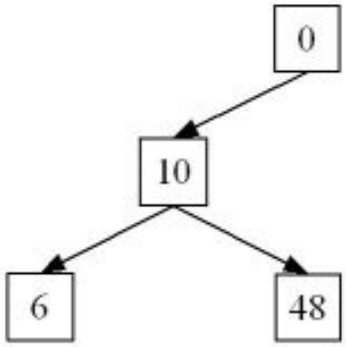


Figure 5

The next number to add is 14, so again recursive calls are made to the left-most node.

Step	R	Size	YourVal	nLeft	NRight
11		7	14	3	3
12	1	3	14	1	1
13	2	1	14	0	0
14	3	1	14	0	0

Value 14 is compared with the left-most node value of 6. 6 is less than 14, so no swap occurs. The function returns to the previous level and 10 and 14 are compared, again 14 is still the greater value so no swap occurs. A recursive call is then made to the right, 14 is less than 48, and the two values are swapped. The value is then returned right up to the root node and assigned. The tree now looks like Figure 6.

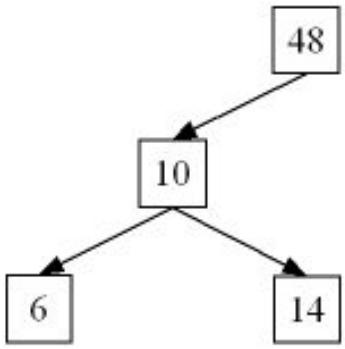


Figure 6

The next number to add is 4, so again recursive calls are made to the left-most node.

Step	R	Size	YourVal	nLeft	NRight
15		7	4	3	3
16	1	3	4	1	1
17	2	1	4	0	0
18	3	1	10	0	0
19	4	3	48	1	1
20	5	1	48	0	0

Since 4 is less than 6, the two values are swapped; 6 is also less than 10, so the values are swapped, and 10 is less than 14 so the values are swapped. The recursive calls return to the top level or root node of the tree and 14 is swapped with 48. A recursive call is made to the right and a new node is allocated with no value assigned. As the process is not "done" and `nLeft` is 1 another recursion is made to the left and the new node is allocated with no value assigned. As the process is still not "done" the value 48 is then assigned to this left-most node of the right-hand side of the tree (Figure 7).

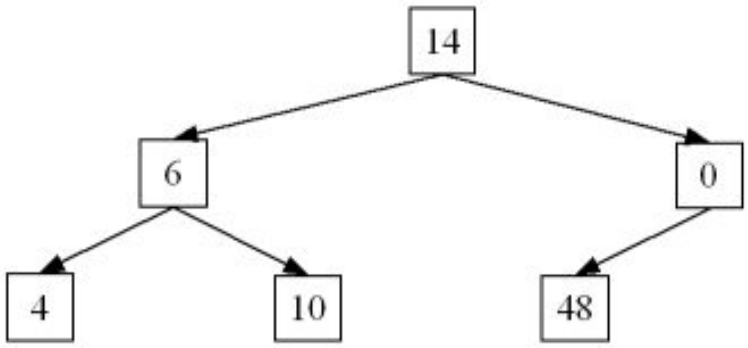


Figure 7

The next value added to the list is 46:

Step	R	Size	YourVal	nLeft	nRight
21		7	46	3	3
22	1	3	46	1	1
23	2	1	46	0	0

24	3	1	46	0	0
25	4	3	46	1	1
26	5	1	46	0	0

Again recursive calls down to the left-most node are made. 46 is greater then all the values stored in the left hand side tree though, so on returning back to the root node, recursive calls are made down the right hand side of the tree and value 46 is swapped with 48. The value 48 is then assigned to the level above, as shown in Figure 8.

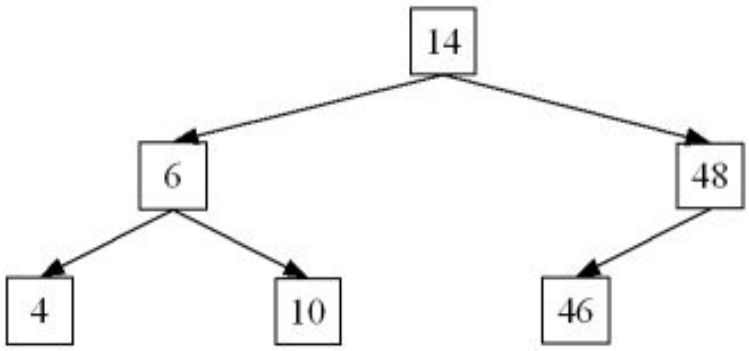


Figure 8

The last number to be added is 21:

Step	R	Size	YourVal	nLeft	nRight
27		7	21	3	3
28	1	3	21	1	1
29	2	1	21	0	0
30	3	1	21	0	0
31	4	3	21	1	1
32	5	1	21	0	0
33	6	1	48	0	0

Since 21 is greater than every number is the left hand side of the tree it is swapped with 46, and 46 is swapped with 48. Another recursive call is made to the right, and the new node is created. As the process is not complete, 48, as the highest number, is assigned to the right-most

node of the tree, as shown in Figure 9.

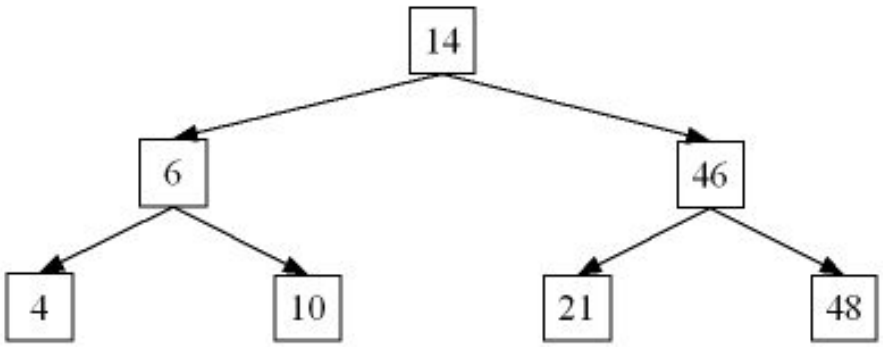


Figure 9. A weight balanced binary search tree

And now I have a Weight Balanced (or Perfectly Balanced) Binary Search Tree.

Summary

For me, Balanced Trees are the point where Computer Science starts to become exciting. The option I've taken will not be the optimum solution for all applications, but depending on the application requirements, only you can decide what is. In the insertion I've designed, comparisons occur between every node when each element is added. Remember the value 46 was initially taken down the left side of the tree, but ended up in the right sub-tree. You may be able to find some way around this, like checking to see whether `yourVal` is greater or less than the current node, and letting this comparison make the decision on whether a move to the left or the right is applicable, however as zero is going to be less than anything I add to the Tree, if I compare the values the zero nodes are always going to be lower than the new value.

In the introduction I said that there are two approaches that can be taken to make sure that the Binary Search Tree always provides the best-case scenario. I've shown how to implement one of these, the Weight Balanced (or Perfectly Balanced) Tree. In my next article I will cover Height Balanced (AVL) Trees.

Download the source

Alison Neal has been using Clarion since 2000, whilst working for Asset Information Systems (AIS) in Auckland, New Zealand. Some years ago (at the tender age of 19) Alison graduated from the Central Institute of Technology in Wellington, New Zealand with a major in Cobol. She also has a BA in English literature and has studied Computer Science, Philosophy and Information Systems. AIS is an independent division of Asset Forestry Ltd, and has a team of five programmers developing almost exclusively in Clarion. AIS also offers web (ClarioNET) and email services for the customer who needs everything. The company has many

and varied customers bridging across a wide range of industries including Telecommunications, Forestry & Agriculture, Manufacturers, Military & Government, Legal & Financial, and Retail.

Reader Comments

Add a comment

Clarion Magazine

Subscribe To Clarion Magazine

Published 2001-01-01

IMPORTANT: Clarion Magazine is an online magazine only - it is not a print publication. For more information, please read the FAQ.

NOTE: If you have already registered a free membership, or have previously subscribed, click here to go directly to the order page.

How To Subscribe

Subscribe To Clarion Magazine Now

Clarion Magazine subscriptions are US$45 for six months, $80 for one year, or $150 for two years. Back issues are an additional purchase, and are discounted.

Although back issues are discounted when you buy them specifically, you can also apply any future issues (we normally publish 48 weekly issues per year) toward any back issues. If you have a current subscription, and try to read an article from an issue you don't yet have, you'll be presented with a one-click option to exchange a future issue for the back issue. That means that for as little as US$45, you can selectively read articles published at any time since Clarion Magazine began publication in February of 1999.

Subscribe To Clarion Magazine Now

How to become a Clarion Magazine member

If you're not already a member, join now. Membership is free, and at a minimum all you need to provide is a user ID and password, your name, and your email address. Members get access

to selected free articles. You also need to become a member to get weekly email updates and other services.

Register Your Free Membership

Why subscribe?

The majority of information in Clarion Magazine is *only* available to subscribers. With well over one million words of Clarion-related information, the equivalent of more than half a dozen Clarion books, the Clarion Magazine web site is the web's best resource for Clarion developers. Here are just a few of Clarion Magazine's features:

- Searchable article database - full boolean search, including title and author search
- Searchable news database - full boolean search, including title search
- Site map
- Topical index - a great way to find articles covering a specific subject
- Links to related articles - each article is linked back to the topical index, letting you quickly find related information
- Downloadable monthly PDFs (plus weekly PDFs in the current month)

For more information see the FAQ, or email subscriptions@clarionmag.com.

Refund policy

Clarion Magazine offers pro-rated refunds if at any time you're not happy with your subscription.

Reader Comments

Add a comment

Clarion Magazine

Clarion News

Published 2001-11-21

Web-Based Clarion Forum

There is a searchable web-based Clarion forum at Tek-Tips. A small FAQ is also available.

Posted Monday, September 30, 2002

XP Menubar Template & Classes Release 1

Andrew Finn has released version 1 of his XP Menubar template and classes. The following features are not yet implemented but are expected soon: Display just hot (color) icons; Use or not use built-in standard icons; Select icons instead of renaming them to allow the template to find them; Hide and unhide menu items and have the template refresh itself.

Posted Monday, September 30, 2002

Free XLib-Based Tool

XScan is a free tool to view, edit and manipulate any data files recognizable by Clarion. XScan is based on XLib library and is able to handle files/tables with structures defined at runtime with a definition script written in Clarion language. Features include: Editing; Mass updates/inserts/deletes; Data migration/export/import from any format to any format; Ability to parse Clarion sources to obtain declarations; JOINed VIEWs, the files/tables can belong to different databases or drivers; Multi-synchronization of several files/tables.

Posted Monday, September 30, 2002

Sterling Data Holiday Schedule

Sterling Data's office will be closed from September 26 through October 8, 2002 for vacation. Mike will be checking his email when possible, but responses will not be as fast as usual.

Posted Wednesday, September 25, 2002

Freeware Calendar

Icetips Software has released its freeware Icetips Calendar. This is not a template, but a simple

calendar program.

Posted Wednesday, September 25, 2002

All Icetips Products 20% Off

As of September 26, 2002 Icetips Software has been in business for one year. To celebrate, Icetips is offering a flat 20% discount on all products for the rest of September, 2002.

Posted Wednesday, September 25, 2002

gFileFind On Sale

Buy gFileFind this week and save $10 (Reg price $79.95 on sale for $69.95).

Posted Wednesday, September 25, 2002

Gitano Weekly Drawing Winners

Last week's winners in the Gitano draw are Steve Ledbetter (gCalc), Steven Wong (Product Scope 32 PRO, Spreadsheet Version), and Adrian Santarelli (Secure Address Book). Prizes this week include PowerSearch, Product Scope 32 Pro, Spreadhseet Version, and XP Firewall Reporter.

Posted Wednesday, September 25, 2002

TPS.repair Templates Price Increase

Effective September, 29th, the price the TPS.repair templates will go up from US$50 to US$75. Trial version available.

Posted Wednesday, September 25, 2002

TX Text Control ActiveX v10 Released

The Imaging Source Europe has released the TX Text Control ActiveX version 10. New features include: PDF Export, which enables you to create PDF files directly from TX Text Control; New button bar with Windows XP look and feel; Updated HTML filter with style sheets; Image export (JPEG and PNG). Klarisoft is an official reseller for The Imaging Source Europe.

Posted Wednesday, September 25, 2002

Icons & Resources - A free development project

Ville Vahtera and Mark Goldberg have put together a resource organizer for Clarion developers. Initially there are seven libraries with around 700 different XP icons. You are not allowed to sold this application as your own! You are free to study what it does, free to change it and free to publish improved versions in the spirit of the GPL. At the beginning of the source you will find email addresses for Ville and Mark's so you can send in any changes.

Posted Wednesday, September 25, 2002

Clarion/PHP Announced

SoftVelocity has officially announced Clarion/PHP 1.0. The release is "just around the corner." With Clarion/PHP you can deploy on Apache (including Linux or Unix versions) or IIS. The price will be the same as for Clarion/ASP, but there will be a special ASP/PHP bundle price.

Posted Friday, September 20, 2002

Clarion 5.6 Threading Model Q&A

The latest SoftVelocity newsletter includes a Q&A section on the new pre-emptive threading model in 5.6.

Posted Friday, September 20, 2002

XP-Like Menu Bar

A beta of a template and classes to create an XP-like menu bar is now available for download. This is the Beta C build so those of you that have previously downloaded the earlier builds should uninstall them first.

Posted Thursday, September 19, 2002

Demo Clarion Web App

José Matzke of Binarianet has a demo of a web application developed entirely in Clarion 5.5f. Criticisms and suggestions welcome.

Posted Thursday, September 19, 2002

ImageEx Special

Until the end of September 2002 you can get ImageEx for US$79, a savings of $20. ImageEx is a set of classes (DLLs) and templates that enhance Clarion's graphics abilities. The four major classes include: The extended image control, which allows your application to display files of many different file types, including TIF, PNG, PCD and a dozen more; The viewer control, which brings ACDSee-like image viewing to your applications with zooming and panning - fast and flicker free; The thumbnailer, which allows your Clarion applications to create thumbnail images; The panner control, which displays panoramic images (with clickable hotspots, tooltips, etc.) Attention existing customers: If you paid more than 79,- US$ for ImageEx, you'll get a discount of the difference price on the purchase of any of other solid.software products.

Posted Thursday, September 19, 2002

Gitano Weekly Drawing

Last week's winner of gBuddy is Rick Martin. This week Gitano Software is giving away gCalc, Encourager Software's Product Scope 32 PRO (single user license); Brady & Associates' Secure Address Book.

Posted Thursday, September 19, 2002

xAppWallpaper Manager v1.6 Released

xAppWallpaper Manager v1.6 is now available. New features include: New method and property to check application's ReadOnly property; of the class and property for check of attribute ReadOnly of the application were added (for starting application from CD); Small fix in template - the rule of definition of name of configuration file is corrected; New demo and install.

Posted Thursday, September 19, 2002

BoTpl Version 2.2

A new version of BoTpl is now available. Changes include: Corrected problems with AppSpecsControl in standalone mode on C5 and earlier versions; W2K defaults to 1.0.0.0; XP defaults to 1.0.0.0 but has additional file version in list that displays correctly.

Posted Thursday, September 19, 2002

ComCode Price Increase

The price of ComCode has increased from US$599 to US$799. Demo available.

Posted Thursday, September 19, 2002

Extended Evaluate Released

Extended Evaluate is a library that allows you to nearly double the execution speed of EVALUATE expressions. Demo available.

Posted Thursday, September 19, 2002

DynaLib 3.0

DynaLib 3.0 is now available. Changes include: Full support of THREAD-attribute for all structures of library; New class TDynaViewClass for creating of dynamic VIEWs of any complication at runtime; New methods in the TBaseGroupClass; New mode of applying of methods DeepAssign; Increased speed of execution in loops; Identity check of two structures; Turn THREAD mode on or off for structure at runtime; and much more.

Posted Thursday, September 19, 2002

Paywire Templates

The Paywire templates are now available, for both ABC and legacy. These allow EFT, ACH and CC processing for your customers and business, (realtime CC processing coming soon). The templates are free, but processing must be done via Paywire.

Posted Thursday, September 19, 2002

Skin v1.003b, Documentation, New Examples

Release 1.003b of iAlchemy's Skin is now available. New features include: Template automatically sets palette to 256 and 0{prop:text} = ''; Examples include a C5PE version ands a handcoded project; Documentation has a step-by-step guide in PDF format.

Posted Thursday, September 19, 2002

Accviz Office

Accviz Office is a productivity suite written by Clarion developer Peck (though not written *in* Clarion). The suite includes: Accviz Word, a general word processor with spell check and thesaurus; Accviz Spreadsheet, a fully featured spreadsheet program. Benefits of the Accviz Office suite include: does not use the registry; does not put any DLLs into the \system32 or \system directory; consumes minimal system resources to run; has a small nag on startup, but do not have any time-outs.

Posted Wednesday, September 11, 2002

BoTpl 2.1 Update

BoTpl version 2.1 is now available. This release includes a fullname feature with checkbox so default View and Print functions will always find the file. Can be turned off for older versions of Clarion not supporting this feature. Also in this release, a variable destination filename has been added to BoAppInfo.

Posted Wednesday, September 11, 2002

GeaSoft Clarion Downloads

Andre Rodella has a number of Clarion source and text downloads available, including DAT file specifications, FTP, POP, and more.

Posted Wednesday, September 11, 2002

Freeware Functions Library Updated

New functions in Frank Uhlik's freeware function library include: HexToDec() - Converts Hex to Decimal; DecToHex() - Converts Decimals to Hex; Toggle() - gets or sets NumLock, CapsLock and ScrollLock; FolderPath() - Strips a filename from a string and returns the path with a trailing backslash. Modifications include: Exist - now works with files and folders, and returns the file attribute value; Kill - now works with wild cards and deletes multiple files (*.* etc). Prototypes have not changed.

Posted Wednesday, September 11, 2002

Financial Query Scripting Language Update

The Next Public Beta for FQL will take place at the end of Next Month. This timetable may be brought forward if beta testing is finished by the end of this month. The New Release of the XI Rules Application Browser (including FQL) adds the following options: A fully re entrant DLL application Model / hosted by any MS Products (VB/C++/DOT NET) and even Clarion; The current host will be in VB EE 5; The Host is written in VB and the Application Browser is written in Clarion. This version supports the Rules scripting object engine. At least 12 new Rule commands have been added including the ability to store business rules on HTTP Servers. This enables FQL to act in the same manner as a WEB Browser delivering

applications and data over a local network or the internet. Also included: Call back embedded supported for VB script Code; Call back embedded support for calling Active X Objects; Database Call Back support to VB / DOT NET so developers can provide there own database record support using any tool they wish to the client Application Browser session; Auto Log on and generation of Database Definitions for MS SQL; The Source code for the VB Host and MS SQL log ons and schemas is included with the Gold Versions of the new Application Browser, so developers can create additional services and call them from FQL. Research has been completed and tested for Mouse objects in the Window object manager allowing control selection and creation/code generation while windows are running in the application Browser.

Posted Wednesday, September 11, 2002

IAlchemy Free Skins

A selection of free skins are available to registered users. Please use same password as your registered copy. More Skins will be added soon.

Posted Wednesday, September 11, 2002

ProDomus Freeware

The freeware Class Removal Template has been modified to remove new methods from Translator Plus in multi-DLL apps not using the library. This template also can removed other classes from an application.

Posted Wednesday, September 11, 2002

Translator Plus Update

An update to the C55 version of Translator Plus has been posted to www.prodomus.com. This Version, 55-09, adds a translator property to turn off translation. It also modifies the optional initialization class to enable loading the last language and locale selection without asking for a selection. The Initialization selection window has been moved to a translation (.trn) file to make editing easier. A progress window has been added to the translator classes load method. This window has also been put in a translation file. Hook handling has been modified to work with CapeSoft's Message Box Template.

Posted Wednesday, September 11, 2002

BoTpl Free Template Release

Comsoft7 is releasing Version 2 of its BoTpl template. BoTpl(tm) is a Free Template set with four extension, one control, and three utility templates, including: Bo_resc(tm), which adds Version Information Resources to the EXE or DLL; AppSpecsControl(tm), a Window control to list information on the build; OneTimeCodeNewBuild(tm) for the frame (with embed) that runs once on new install or version update; ProcStatBar(tm) puts the name of the active procedure and template type into the status bar of the main frame; BoAppInfo(tm), a utility for information about your app; DicPrint(tm), another dictionary to text/print utility;

EmbedsAllOut(tm) which captures module embeds (no longer supported in the BEAOT.Tpl(tm), but is now part of BoTpl(tm)).

Posted Wednesday, September 11, 2002

Commercial Clarion Apps

This page on the sa/design web site lists known commercial applications created with Clarion.

Posted Wednesday, September 11, 2002

TPS Encryption Tips

Wayne Price offers a number of valuable encryption tips in this message posted on the SoftVelocity news server.

Posted Monday, September 09, 2002

INN Bio for 5-Sep-2002

This week's INN Bio takes goes south and west... and south and west some more. Once a milkman, then a corporate bean-counter, this Clarionite now owns his own software business. When home, he enjoys baking bread with the family and listening to his 400 ??? CDs. His home is located in a place with a few unfriendly critters, and the snide comment about Texas is not appreciated.

Posted Monday, September 09, 2002

Clarion LibMaker 32 Version 2.0 (Freeware)

Clarion LibMaker 32 is an advanced version of the Limaker utility program found in the BIN-folder of Clarion for Windows. This version of Clarion LibMaker 32 includes several new functions previously only seen on the wish-list. Most of the code in the entire program is now rewritten and optimized for speed. Features include: Favorites button with API LIB-files preset; Cleanup functions that removes duplicate symbols automatically; Automatically locates source DLL-files; Updates LIB-files from DLLs on disk automatically; Set default source and target folders; Sort lists from clickable header buttons; Single and multi keyword searches; Compare LIB-files to DLLs/LIBs on disk; Drag & drop single or multiple files; Open single or multiple files from command line; Make LIB-files from within Clarion templates; Many user options

Posted Monday, September 09, 2002

SealSoft Discounts

If you've purchased one or more SealSoft products for a total sum of $75, you're eligible for the SealSoft Personal Discount Card, which entitles you to a 10% discount on subsequent purchases.

Posted Monday, September 09, 2002

New Weekly Drawing

Each week Gitano Software holds a free drawing. Just enter your name and email.
Posted Monday, September 09, 2002

USF By iAlchemy

iAlchemy has launched USF - Undocumented Shell Functions. Features include: Monitor mouse & keyboard inactivity (IdleTime) across the entire system, not just your Clarion application; Execute Reboot, Shutdown, Windows Logoff, and Run as "soft" or "hard" functions; Lock/unlock files. Priced at US$14.99. Demo available.
Posted Monday, September 09, 2002

IconsXP, Theme pack 1, Look Good Package Updates

All icons are now included as GIF files as well, that is 160+ images that you can use in your web apps. The template in the Look Good Package has been modified to add support for windowless procedures. This update is free to all registered users.
Posted Monday, September 09, 2002

BoxSoft Super QuickBooks-Export Templates

BoxSoft has released the Super QuickBooks-Export templates. This product creates IIF files that can be imported into virtually all versions of QuickBooks. For more information, download the templates and run the installation. This will let you read the documentation which is installed before you're prompted for the password. To purchase templates, please contact Mitten Software at Mitten@MittenSoftware.com. Their phone numbers are (800)825-5461 and (952)745-4941.
Posted Tuesday, September 03, 2002

solid.software Holiday Schedule

The solid.software office will be closed from August 31 through September 8 2002 for vacation. All emails will be answered when Jens gets back. Products can still be ordered at ClarionShop.
Posted Tuesday, September 03, 2002

Linder Software Vacation Schedule

The Linder Software office will be closed from September 1st through September 8th
Posted Tuesday, September 03, 2002

Skin 1.002 Released

Skin 1.002 is now available. Changes include: Performance modifications; Minor template alterations; "Suppress frame" on window move. An enhanced demo is also available, which lets you preview your own skins.
Posted Tuesday, September 03, 2002

Clarion Freeware

Sterling Data has a number of free downloads available, including a utility to read Clarion DAT file headers, which can aid in recovering data from trashed DAT files.

Posted Tuesday, September 03, 2002

Anti-Cracking FAQ

This page on Inner-Smile Software's web site details measures you can take to make your software less susceptible to cracking. Well worth a read.

Posted Tuesday, September 03, 2002

Reader Comments

Add a comment

TopSpeed INI Files

by Tim Phillips

Published 2002-09-13

Windows INI files are both a blessing and a curse. They make it trivial to keep track of users' default settings. They can, however, become real trouble if a setting goes wild (ever tried to patch one via a phone call to a user who has never run Windows Explorer or Notepad?) or if you are trying to support users who shift from computer to computer and would like their settings to follow them around the network.

I think INI files need two improvements to be really useful. First, the developer should have control where the INI file is stored, so it can be placed on a shared network drive for users who move between computers. Second, INI files should be a lot easier to update/troubleshoot, i.e. via a set of browse/form procedures inside the application itself.

As it turns out, it's not that difficult to use TPS (or other database files) as drop-in replacements for text INI files in ABC applications. In this article I'll explain how you can extend the ABC IniClass with just 25 lines of code, and drop in your own INI file handler.

> **NOTE**: I was once a hard-core Legacy code programmer, but this is one of those situations where Legacy templates would be an extensive exercise in pain (especially if you use a lot of third party templates as I do). You would have to find all the GetINI() and PutINI() commands through all your templates and rewrite them to be "non-standard". I am sure you could do it, but it wouldn't be a trivial exercise. This is a case where the ABC Templates really shine. The ABC Templates utilize the INI Class. If you can control that class, you can control the behavior of the entire application.

Creating the data file

The first step in this process is to declare a new TopSpeed file in your data dictionary. I called mine `INIFile`, and it holds the INI data. The prefix is `INI` and the Full PathName is set to `!GLO:INIFileName`. The `!` prefix indicates this is a variable, so I also declare a global variable (prefix `GLO`) called `INIFileName`, which is a `STRING(255)`. I prefer the TopSpeed format for this file but there is no reason another format wouldn't work also. Here's the layout:

```
INIFile          FILE,DRIVER('TOPSPEED'),NAME(GLO:INIFile),PRE(INI),|
                      CREATE,BINDABLE
ByISectionIEntry   KEY(INI:ISection,INI:IEntry),DUP,NOCASE,OPT
Record             RECORD,PRE()
ISection             STRING(100)
IEntry               STRING(100)
IValue               STRING(100)
                   END
                 END
```

The `INIFile` layout mimics that of an INI file, which has multiple sections. Each section has one or more entries, and each of those has a single value. For the file's layout, I declare three fields `ISection`, `IEntry` and `Ivalue`, each of which is a string of 100 characters. I also declare a key of `ByISectionIEntry` that is a combination key of `ISection` and `IEntry`.

When I first investigated how to use a TPS file for INI data, I attempted to write the code for my own INI Class. A little reading of the INI class source, however, shows that this is more of a pain than it is worth. Why duplicate and then maintain excess code when all I really needed to do was to override the default behavior of just a few methods? Wait a moment, doesn't this sound like what I do every day with just the proper embedded code points?

A little more digging landed me in the Global Embeds for the application. Here I found embeds for the `INIMgr` under Global Objects, ABC Objects. With these embeds, I was able to override just that code that I wished to, while the parent class remained to handle all the other details (and, of course, that code is maintained by the experts at SoftVelocity instead of by me!).

Here's how to add a TPS database capability to `INIClass`.

Creating a derived class

First, you need to control the name of the file initialized by the `IniMgr` object (so your INI file is stored where you want it to be). Go to the Global Objects, ABC Objects, INIMgr, Init embed and insert the following code above the parent call:

```
If Exists('H:\')=True ! is a private user network drive available
```

```
 GLO:INIFile='H:\system\INIStaffCalender.TPS' ! set file name
Else ! there is no network drive
 GLO:INIFile=CLIP(GLO:WindowsDirectory) & '\INIStaffCalender.TPS'
.
Relate:INIFile.Open
```

What does this code do? Well, the `IF` statement is just attempting to correctly set the `GLO:INIFile` to a valid location. In my office, when users log into our network, they pick up an H: drive mapping to a private area on the server. That area will always have a `\System` directory, which is where the IS staff wants things like stored settings to be placed. If there is no H: drive, the program is probably being run from a laptop outside our offices, so the proper place to put the settings is into the Windows directory (the `GLO:WindowsDirectory` variable is set ahead of time using the appropriate API call to get the Windows directory of the PC).

The `Relate.INIFile.Open` method serves to open the `INIFile` so that the `INIMgr` is ready to save/retrieve data from the file.

Next, you need to write the code to put information into your new INI file. Go to the Global Objects, ABC Objects , `INIMgr` , `Update PROCEDURE(STRING Sector,STRING Name,STRING Value,STRING Filename),VIRTUAL PROTECTED` embed and add the following code above the parent call:

```
INI:ISection=Sector
INI:IEntry=Name
ifAccess:IniFile.TryFetch(INI:ByISectionIEntry)=Level:Benign
 INI:IValue=Value
 Access:IniFile.Update()
else
 INI:ISection=Sector
 INI:IEntry=Name
 INI:IValue=Value
 Access:IniFile.Insert()
end
Return
```

This code attempts to `FETCH()` a match for the Sector-Name combination. If a match if found, `Value` is updated. If no match is found, a new INI record is inserted to hold `Value`. The `Return` statement will cause an exit from the procedure before the Parent call; this prevents the `IniMgr` default code in the procedure from using the "original" INI file architecture.

Next, you need to write the code to fetch information from the new INI file. Go to the Global Objects , ABC Objects , `INIMgr` , `Fetch PROCEDURE(STRING Sector,STRING Name,STRING Default, STRING Filename),STRING,VIRTUAL PROTECTED` embed and add the following code above the parent call:

```
INI:ISection=Sector
INI:IEntry=Name
if Access:IniFile.TryFetch(|
   INI:ByISectionIEntry)=Level:Benign
 Return INI:IValue
else
 Return Default
end
```

This code attempts to `FETCH()` a match for the Sector/Name combination. If a match is found, the `Value` is returned. If no match is found, the `Default` value is returned.

Now, there should only be two steps left to make the new code functional. If you do not have the Global checkbox for Generate All File Declarations turned on, you need to go to the Individual File Overrides, find the entry for the `INIFile` and turn the Generate File Declaration Property on. This will make sure that the `INIFile` is declared properly so you can use it as early as when the `INIMgr` is initialized.

While you are on the Individual File Overrides tab for the `INIFIle`, set the THREADED value under FILE ATTRIBUTES to Not Threaded. If you do not do this, you will probably generate a whole pile of runtime errors about the `INIFile` not being open as the program runs. This error appears to be tied to the known errors with threading, so hopefully, Clarion 5.6 will fix it. Until then, setting this file to NOT THREADED is a workaround that has served me for many months.

The application should now compile. When it runs, the former functionality of the INI file is now taken by whatever `GLO:INIFile` is pointed at.

Now that you have the data in a usable format, you are half-way home! The next step is to use the data.

Creating a browse/form combination for the `INIFile` is as simple as using the wizards and tuning (or importing the `TPSINIbrowseform.txa` file included with the downloadable source). The only "unusual" thing I did was add a Record Filter of `CLIP(INI:ISection)<>'__Dont_Touch_Me__'` to the browse I created. This will "hide" the INI File line that shouldn’t be disturbed, so there is no way it can be deleted or changed from the browse.

You can call the `INIFile` browse/form combination from a Menu item or a button on a User Preferences window or whatever makes sense.

You now have a "normal" browse/form way to maintain the data in the INI file. As soon as you do this, however, you will probably rapidly realize that you might not *want* to do this.

A number of variables in the INI file (Preserved globals, the window size for the Main frame..) are things that you could change from the `INIFile` browse/form combination, but they will be overwritten when the preserved globals are saved as the application shuts down or when the window in question is closed and its location is saved to the INI File.

This will, at minimum, confuse the user. At worst, the values (preserved globals, window size of main frame) you really can't control from inside the application may literally be the values you want to overwrite and control via the INI File.

You basically have two options

1. Accept that certain variables can't be controlled from a browse/form located inside the application. I would then recommend applying appropriate filters to the internal browse so that these records don't even appear in the listing.
2. Move the browse/form combination outside the application itself.

Option 2 is more in keeping with how you generally would work on an INI file anyway; you would shut the application down first and then make your changes. To handle this, you need a small application (an applet in my terminology) that does nothing but provide a browse/form to handle the INI file.

You can fabricate this yourself, or you can import `tpsiniapplet.txa` into a blank APP file to form the application. The only "cute" thing that I've done with this application is put the line `GLO:IniFile=Command('')` in the Local Objects, This Window, Init, Initialize the Procedure embed. This lets you specify the contents of the `GLO:INIFile` variable (which defines what your INI file is called) with command-line parameter. This makes it easy to control which file the TPS INI applet is trying to work with.

With the applet compiled, you can use it in three ways.

1. From within the parent application itself, you can call the TPS INI Applet. I use a statement like `Chain(('O:\DECUtil\TIapplet.exe ' & CLIP(GLO:INIFile)))` on an INI FILE button on a User Preferences Window. The `CHAIN()` command will cause the parent application to shutdown and then activate the INI Applet. Some would say this is risky (for instance, if the user has a form open, unsaved data will be lost when the `CHAIN()` is started). It does, however, keep everything self-contained. What may work for some is to have a warning window with a cancel option that lets the user know any unsaved data will be lost.
2. Tou can install a shortcut along with your main application on the user's desktop. This would turn on the INI Applet with a TARGET line that would look like `"O:\DECUtil\TIapplet.exe" "H:\system\INIStaffCalender.TPS"`. This technique means that user isn't automatically/inappropriately shutting the main

application down to view the INI file, but it does mean that the user has to manually shut the main application down in order to successfully change all the fields in the INI file. This technique, is, however, almost identical to what you would tell a user to do with "normal" INI files.

3. You can create a shortcut in the SENDTO directory under Windows on your PC. If you set the TARGET to something like `"O:\DECUtil\TIapplet.exe"`, you can navigate to various Topspeed INI files on your network (reaching out to a confused user on the other side of the complex, say) and use the SEND TO option in Windows Explorer to send their INI file to the TPS INI Applet. You can then modify the file yourself remotely from your own desk. The `COMMAND()` statement in the applet will pick up the file name you are using SEND TO with from the operating system and use it to correctly set `GLO:INIFile`.

So there you have it. A handful of lines of code and a simple browse/form combination and you can convert "old style" INI files into an architecture that you can actually work with as easily as you do the rest of your database files. Pull off a couple of stunts like this, with so little code written and effort expended, and ABC can get to grow on you.

Download the source

Tim Phillips began programming Clarion with Version 3.0 for DOS. He currently works mainly with Clarion 5 but is determined to get everything into 5.5 by 2002. His preferred programming technique involves a lot of bass rock guitar on his cordless headset and vigorous application of the Keep It Simple Stupid principle. When not programming, he can be found trying to write fiction, "building Legos" with his two nieces, or watching too much quality television.

Reader Comments

Add a comment

Tim, very clever and interesting use of resources....

Limerick Contest Results

Published 2002-09-13

Here are the results of the Clarion Magazine Limerick contest. Although we received only three entries, all were of good quality, and it was difficult to choose the winner. The rules were as follows:

- The limerick itself must be compilable Clarion source code, in the form of an IF, LOOP, or CASE structure.
- Words to be used can be declared (apart from the verse) as equates, data labels, calls to external routines, etc.
- Points given for actual usefulness of the code, rhyme, meter, double-entendre (but please remember this is a family magazine), etc.
- Line continuation characters are considered punctuation. You may only use a comment on the last line.

Randy Rogers, inspired by George Cobaugh, submitted the following limerick:

```
  program
  map
Chat        procedure()
Spend       procedure(*long Money)
  end

TaxDiscussion window
              end

There_Is_Money_My_Friend    long(100)
Money_For_Foods             long(10)
Tax_Me_Again_And_Again      byte(true)
Demand                      equate(0)
Jobs                        equate(0)
For_The_Goods               equate(create:string)
For_People_Who_Spend        equate(create:string)

  code
```

```
    Chat()

Chat    procedure
  code
    open(TaxDiscussion)
    ! thanks to george cobaugh for the inspiration
    loop while There_Is_Money_My_Friend |
    and Tax_Me_Again_And_Again
      spend(Money_For_Foods)
      create(Demand,For_The_Goods)
      create(Jobs,For_People_Who_Spend).

    close(TaxDiscussion)

Spend   procedure(*long Money)
  code
    There_Is_Money_My_Friend -= Money
```

Our judge detected a slight slip of the meter in "`create(Demand,For_The_Goods)`", and a wee bit of poetic license in rhyming friend, again, and spend. Also, it's a while since our judge bought "foods." Still, very nicely done, and probably as good a model of economics as any other. Top marks for the code actually doing something.

Steven Gallafent not only supplied a contest entry, he introduced it with a limerick:

Here my entry for you to peruse.
It compiles, but that's isn't news.
If it's answers you seek,
Then in English, not Greek,
This program should help you to choose!

Steven also points out that in his entry the = signs are pronounced "equals," and all of the other punctuation is just there for the compiler.

```
! Steve Gallafent
! steve@compguy.com
  PROGRAM

  MAP
    TellThem(STRING)
  END

  ITEMIZE
IngWhoWins         EQUATE
Tough              EQUATE
ReadingTheseRhymes EQUATE
Rough              EQUATE
  END

YourPencilDownNow  &FILE
ThePrize           &FILE
```

```
ToMyHouse          STRING('I never get sent')

  CODE

  ! Limerick starts here
  IF CHOOSE(IngWhoWins = Tough) |
     OR (ReadingTheseRhymes = Rough)
    PUT(YourPencilDownNow)
    SEND(ThePrize,ToMyHouse)
  END ; TellThem('We''ve read quite enough')
  ! Limerick ends here

TellThem           PROCEDURE(STRING WhatToTell)
  CODE
  MESSAGE(WhatToTell)
```

Our judge gave extra marks for "`CHOOSE(IngWhoWins)`" and the creative use of the semicolon to get around the `END` problem, but "now" doesn't rhyme with "house," at least around here.

Finally, **Mike Ware** sent in a "classy" limerick. I guess you could say he, ah, typed it:

```
LimerickType Interface
MyTime         PROCEDURE (BYTE Waist)
 .
PoetryHype   CLASS, DLL, TYPE
Language       Procedure(*Group Taste)
 .

MyTry CLASS(PoetryHype)|
implements(LimerickType)
LineCount   BYTE(5)
Words       ANY ! (Derive)
Output      PROCEDURE (? bagpipe)
       .
```

As with the best object-oriented code, this limerick is elegant in its simplicity. There are several problems here also, however. The code doesn't quite fit the requirement of code in the form of an `IF`, `CASE`, or `LOOP` structure, and it uses a comment but not on the last line (where it was intended to help with the `END` problem). On the other hand, there is only one comment, and it makes sense in the context.

Although this entry misses a couple of requirements, the other two entries don't quite fit the limerick standard for rhyme and/or meter.(Actually none of the entries quite fit the classic limerick structure.) There is no completely qualifying entry.

Exercising the magisterial prerogative, our judge gave extra points for thinking outside the box, and designated Mike Ware the winner. Congratulations, Mike! Richard Rogers has been notified, and you will be receiving your prize.

Reader Comments

Add a comment

Clarion Magazine

Frequently Asked Questions

Published 2001-08-20

What kind of magazine is this?

Clarion Magazine is an electronic magazine, published on our web site (http://www.clarionmag.com). The articles in Clarion Magazine are of interest to software developers who use the Clarion language from SoftVelocity, Inc.

How often is it published?

Clarion Magazine is published on an ongoing basis, but for subscription tracking purposes there are 48 weekly issues per year. New articles are posted throughout the week, as are news items.

How much does it cost?

Subscription rates are US$45 for six months (24 weekly issues), US$80 for one year (48 weekly issues), and US$150 for two years (96 weekly issues). Although Clarion Magazine is an online publication, the subscription model follows that of a print publication. That is, your subscription gets you all issues published from the start of your subscription forward. You can exchange future issues for back issues online (if you attempt to read an article from a back issue you don't have, and you have unused issues, you'll be presented with a one-click option to exchange issues and read that article).

You can also purchase all the back issues at the same time as you purchase your subscription, or at any other time. There's no fixed price for the back issues as they are constantly accumulating (ClarionMag began publication in February, 1999), but they are somewhat discounted. The subscription purchase page will calculate the cost of any back issues you don't yet have. If you want all the back issues, it's definitely less expensive to buy them rather than

exchange future issues, for which you pay full price.

You always have access to the issues you've purchased, even if you don't have a currently active subscription.

Are some pages freely available?

All of the Clarion Magazine general information pages, news pages, and occasionally individual articles are available at no charge. To access some pages you may need a free membership. All of the Clarion Online articles are also available at no charge. You may also want to download the ClarionMag Free Sampler PDF.

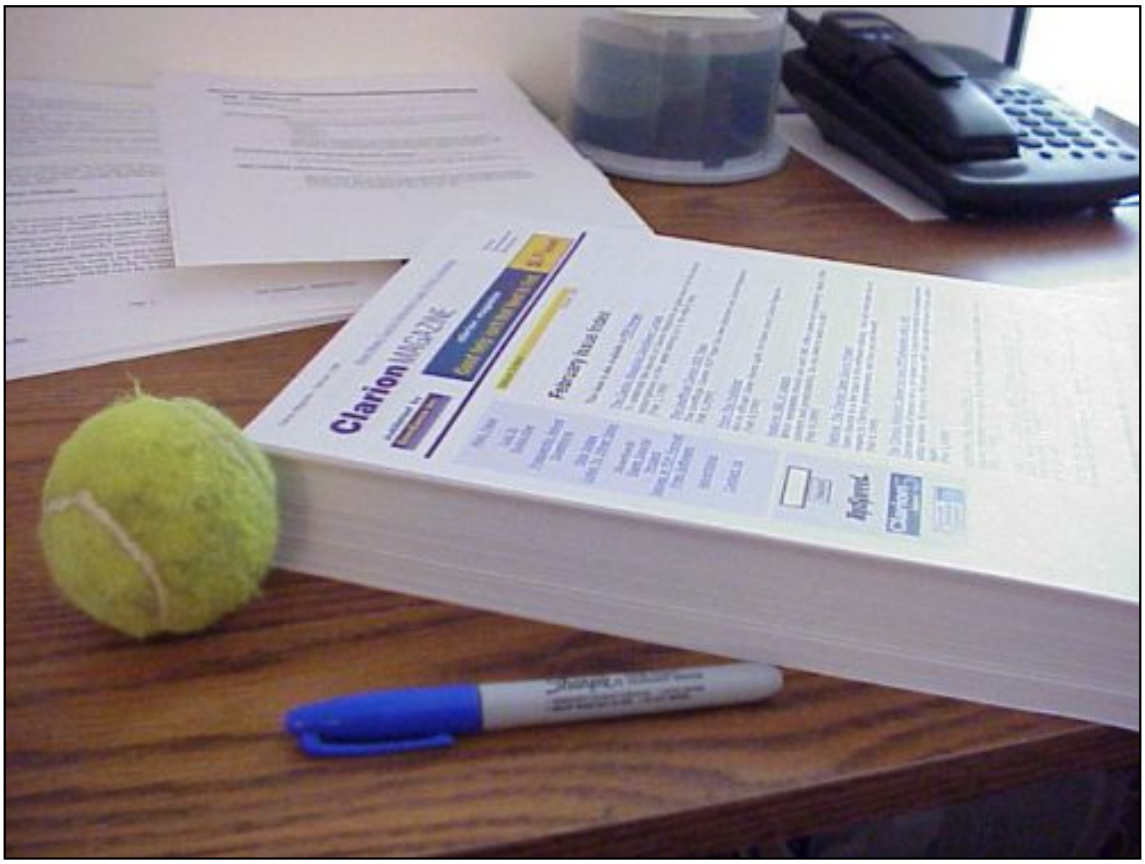
Why should I subscribe?

One of the cardinal rules of software development is don't build what you can buy. You wouldn't think twice about spending the cost of a subscription on a third party product that saved you hours of work and helped you provide a better application to your customers. And that's exactly what Clarion Magazine is: a product that saves you time and improves the quality of your work. You get how-to articles, source code, tips and techniques, product reviews, Clarion news, and more. You can even exchange comments with authors and other readers via the reader comment link at the bottom of most article pages. As with any other good quality third-party product, Clarion Magazine offers a terrific return on investment.

How much information does Clarion Magazine contain?

As of August, 2001, the entire www.clarionmag.com site contains over one million words of Clarion-related information. The majority of that information is made up of articles published in Clarion Magazine; the rest is in the freely available Clarion Online archives. That's the equivalent of six fairly large computer books, and it's all available through index pages and the search page.

What would that look like in hard copy? The following picture shows Russ Eggen's printout of the *1999 articles only*, double-sided (thanks, Russ!). That's a little over a ream of paper (500 sheets, or 1000 pages). If Russ printed everything on the Clarion Magazine web site, as of September 2002, that would be a stack of paper printed on both sides, over a foot high!



The 1999 Clarion Magazine articles, duplex printed.
(Photo courtesy of Russ Eggen)

Can I advertise in Clarion Magazine?

Definitely! With over 50,000 page views per month, Clarion Magazine is a great place to advertise your products and services to other Clarion developers.

What's the connection between Clarion Online and Clarion Magazine?

In 2000, after Clarion Online ceased publication, Clarion Magazine entered into an agreement with the publisher of Clarion Online to make that magazine's articles available on the Clarion Magazine web site, at no charge to readers. The transfer of copyright which made this possible is the only connection between Online Publications Inc, which published Clarion Online, and CoveComm Inc, which publishes Clarion Magazine.

Is Clarion Magazine available in print form?

As a paper journal, no. However Clarion Magazine is available in PDF form which prints very nicely. You can download the February 1999 issue in PDF format if you'd like to see how it works. You can also print the pages from your browser - most pages have a printer-friendly link which removes the left side menu and right side links.

Is the February 1999 issue a good indication of what I can expect?

You can expect a bit more, actually. The February preview ran for three weeks/three issues, rather than the more usual four issues per month. As well, some of the information posted in February was about the magazine itself, and not necessarily about Clarion programming. You'll see more source code and real-world examples in the articles you read as a subscriber.

Who is behind Clarion Magazine?

Clarion Magazine is published by Dave Harms, president of CoveComm Inc, a Clarion consulting company. Dave is the co-author with Ross Santos of the book Developing Clarion for Windows Applications (SAMS 1995). He is also one of the principal contributing writers to Clarion Magazine. A Clarion developer since 1990, Dave is a frequent conference speaker. His Clarion work has taken him from his home in Canada to numerous locations in the US, and to Italy, Singapore, and Australia. He is the co-author of the books Web Site Programming With Java (McGraw-Hill, 1996) and JSP, Servlets, and MySQL (HungryMinds, 2001), and has a background in magazine and book publishing as well as writing.

Dave is assisted by a number of highly-qualified writers and by an advisory board made up of some of the leading figures in the Clarion community. The board helps ensure that Clarion Magazine stays true to its mission of being the best single-point source of Clarion information anywhere.

What if I don't like the magazine?

In that unlikely event, prorated refunds are available. Read our refund policy.

What if I have a question that's not in the FAQ?

Send it to editor@clarionmag.com!

Reader Comments

Add a comment

I Smell a rat! Russ doesn't play tennis, he plays...

Managing Skipped Field Hot Keys

by Carl Barnes

Published 2002-09-19

If I think a field should be rarely used, or not changed from its default value, I'll typically give it a SKIP attribute so the user will not be asked to enter it while tabbing through the fields. This allows faster data entry and users like that. It also keeps default values from getting accidentally hosed. For example, an Invoice date will default to today's date and I think it should be rare that it needs to be changed, so I want it skipped. The Window code with the SKIP attribute on the ENTRY does this:

```
PROMPT('Invoice &Date'),AT(147,114),USE(?PRO:DatePrompt)
ENTRY(@d2),AT(203,112),USE(PRO:InvoiceDate),SKIP
```

If the users want to override the default and enter the Date they can press the hot key designated in the PROMPT, which is Altd+D (&D), right? Sorry, but that will not put focus on the date. The PROMPT will give focus to the next active control. The SKIP attribute on the Invoice date ENTRY will prevent it from getting focus. The only way the users can change the date is by clicking on it with the mouse. My users generally do not like that method; if they have to take their hands off the keyboard it slows them down.

The simple solution is to add a KEY(AltD) attribute to the ENTRY. The KEY attribute allows you to specify a hot key to give focus to the control. In my testing, KEY ignores SKIP. I leave the "&D" hot key in the prompt so the user knows what key to hit. This does not seem to cause any conflicts, although this use of KEY is not explicitly documented. Below is the final code.

```
PROMPT('&Invoice Date'),AT(147,114),USE(?PRO:DatePrompt)
ENTRY(@d2),AT(203,112),USE(PRO:InvoiceDate),SKIP,KEY(AltD)
```

Other Controls

If a separate PROMPT control is used, as is the typical case with COMBO, CUSTOM, ENTRY, LIST, SPIN, and TEXT controls, the hot key only takes you to the prompt; the ACCEPT loop sees the SKIP attribute on the following control anyway, and moves focus to the next control. These controls require a KEY() attribute to give them focus with a hot key if you want them to have the SKIP attribute.

If you're using CHECK controls with SKIP, adding KEY() is not required because CHECKs have the prompt, and therefore the hot key, as part of the control declaration (e.g. CHECK('Check &Me')). This hot key does not allow the control to take focus; instead, it sends it an EVENT:Accepted which toggles the checkbox. This is desirable behavior; the current control retains focus after the checkbox is toggled.

If you do add the KEY() attribute to a CHECK then it will work like the other controls and that hot key will give the CHECK focus (and toggle it). However, there is a bug in the runtime library that causes a CHECK control with SKIP to not paint a selection box around the prompt if it gets focus with a KEY. The CHECK will gain focus, but the user will not be able to see it. No control will appear to have focus. You can tell the CHECK has focus because pressing the spacebar will toggle it. Due to this bug KEY should not be used on a CHECK with SKIP. I have tested this in Clarion 5 and 5.5.

RADIO controls behave exactly like CHECK controls so the above discussion applies to them also; give them a hot key (e.g. RADIO('&Single')) but do not use KEY(). The OPTION control, that groups the RADIO controls, is somewhat like a PROMPT control. It never gains focus (gets an EVENT:Selected) so giving it a SKIP attribute does nothing. To remove an OPTION from the tab sequence you must place SKIP on all of its RADIO controls. If you do not give each skipped RADIO control a hot key the user will have to use the mouse to change it.

At times it will be impossible to find hot keys for all of the RADIO controls. But I still want to allow a way to use the keyboard to select the control. In this cause I do assign the OPTION a hot key e.g. OPTION('&Marital Status'). Because the radios are skipped the Alt+M does nothing, so I need to ALERT the Alt+M key on the Window. When I get an EVENT:AlertKey for Alt+M I change the radio controls to no longer be skipped, and then select the OPTION control so it will have focus. The user can change options with the arrows then press tab. The below code from the window alert key embed shows this technique. See the help for PROP:Child and PROP:ChildIndex to enhance this code to handle any number of options.

```
IF KEYCODE()=AltM
   ?Marital:RadioSiHappy{PROP:Skip} = 0
   ?Marital:RadioMarried{PROP:Skip} = 0
   ?Marital:RadioHeadHse{PROP:Skip} = 0
   SELECT(?MaritalStatusOption)
```

```
    DISPLAY
    CYCLE
END
```

A GROUP control behaves like an OPTION control and so the above discussion on OPTION applies. Placing SKIP on a GROUP does nothing. You must place SKIP on all the controls within the GROUP. See the help on the PROP:Parent property, it makes it easy to find all of the child controls of a GROUP. (I find it somewhat inconsistent but placing DISABLE or HIDE on an OPTION or GROUP does apply to all of the child controls.)

A BUTTON control works very much like a CHECK or RADIO. A button can have a prompt (e.g. BUTTON('&Save')) that provides a hot key. The Alt+S can be used press the button if it has the SKIP attribute. If the button was changed to have an ICON and no longer have the prompt (e.g. BUTTON(),ICON(ICON:Save)) the functionality can be retained by adding a KEY(AltS) so the user can still press Alt+S. In this case you'll have to tell the user about the secret hot key with a TIP or documentation.

KEY can be very useful in adding hot keys that do not use Alt keys to speed data entry; for example, KEY(F10Key) might be assigned to a BUTTON('...') to initiate a lookup. This saves your users from taking their hands off the keyboard to grab the mouse. Read the Help on KEY, SKIP and PROMPT for more info.

These techniques make it practical to add SKIP to many fields while still providing keyboard hot keys. If you read Joel on Software you'll find one of his beliefs is that users want to enter less data, and want developers to make choices for them. I've come to believe that statement. Users do not want lots of options to think about and enter. Call it "Tastes Great" *and* "Less Filling". By using SKIP and KEY you can have extra entries that do not get in the users' way.

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

Reader Comments

Add a comment

Other stupid KEY tricks: If you want to define a...

Clarion Magazine

A Tool For Understanding Template Symbols

by Steffen Rasmussen

Published 2002-09-19

In the article Understanding the Template Symbols I introduced my preferred workflow when creating templates, as well as a technique to determine which template symbols to use. As I mentioned, I use trial and error, placing comments into the template code containing the template symbols I think are usable. For this I have a file with the most used template symbols that I use, and then it's just a matter of copying the content into the code embed where I want to use it. It's not the most elegant approach, but it has served me fine. I considered including this file with the article Understanding the Template Symbols, but then I got the idea to make a code template instead. The code template should include all possible template symbols, and after all it shouldn't be that difficult to create since it is mostly a copy and paste job from the Programmer's Guide.

Copy, Paste, Search and Replace

The overall structure of each template symbol command is:

```
#IF(%TemplateSymbol <>%Null)
!%%TemplateSymbol=% TemplateSymbol     ! TemplateSymbol comment
#ENDIF
```

Unfortunately I don't have any one editor that does everything I want, so instead I used a combination of different programs. First I copied all the template symbols with comments from the Programmer's Guide into a Wordpad document, then I imported it into Excel. When importing the file I made sure that the template symbol command was in its own column so I could copy the symbols apart from the comments. I also created extra lines for the `#IF` statements and added extra columns to display the variables as comments (`!%%'` and `'=%`). Then when the Excel file was finished I saved it as a doc file, opened it in Wordpad, and

copied it into a new tpl file. Next it was just a matter of making a search and replace of all the spaces from the columns that I didn't need. That was all I had planned, but then I realized it would have been nice to include the #FOR structure and as well, be able to select the template symbol groups which I wanted to see, and turn the #FOR structure on and off as needed. *Sum a summarium*, the template became a lot more complicated than I originally anticipated.

So how does the template work?

The template is a code template which you place in the embed point where you want to find a template symbol. The template prompts display a hierarchy overview, as shown in Figure 1.

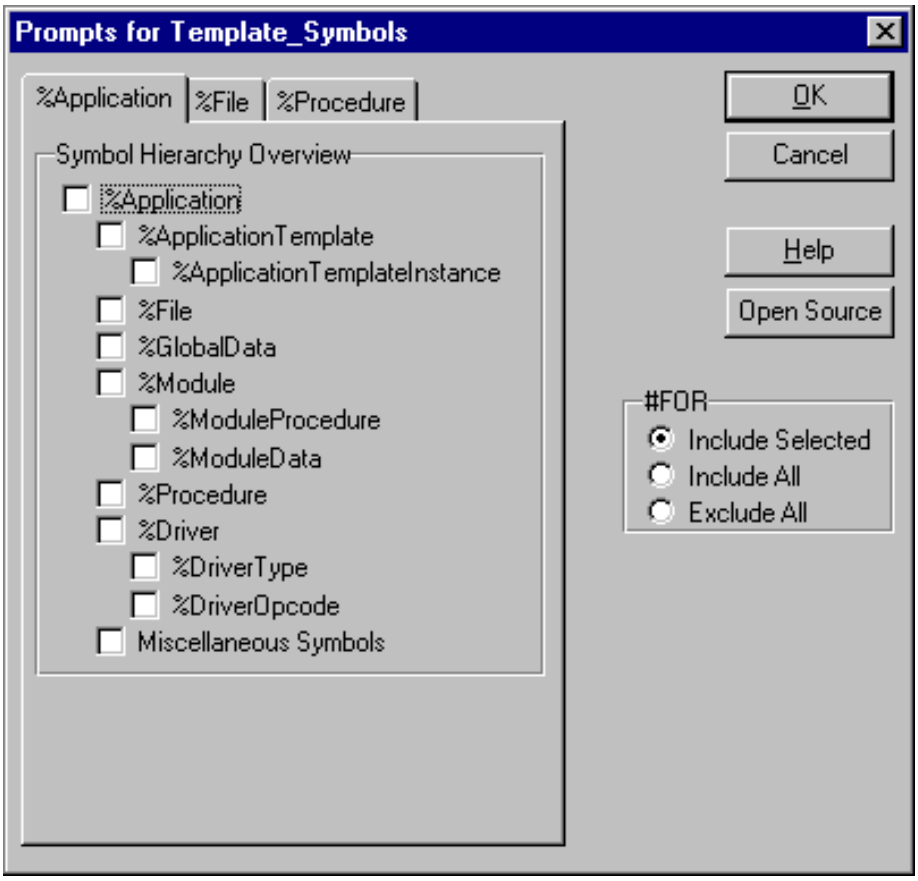


Figure 1. The symbol hierarchy overview

As you can see from Figure 1, the template dialog contains three tabs, which divide the hierarchy into three parts. The first tab shows the main structure of the symbol hierarchy, the second tab shows symbols dependent on %File, and the third shows symbols dependent on %Procedure. The check boxes are placed in such an order so that each column is dependent on the previous column. For example %ApplicationTemplateInstance is dependent on %ApplicationTemplate, which then again is dependent on %Application. %File on the other hand is only dependent on the %Application. Now it is just a matter of checking the template symbol groups that you want to see.

On the right side of the template window there is a #FOR option box containing three radio buttons:

- Include Selected
- Include All
- Exclude All

Include selected means that all the selected template groups will be included within a #FOR loop. For example if you only select %ApplicationTemplateInstance the template code structure will look something like this:

```
#FOR(%ApplicationTemplateInstance)
!%%ApplicationTemplateInstance=%ApplicationTemplateInstance
#ENDFOR
```

On the other hand if you select the Include All option, the template code structure will look something like this:

```
#FOR(%Application)
    #FOR(%ApplicationTemplate)
        #FOR(%ApplicationTemplateInstance)
!%%ApplicationTemplateInstance=%ApplicationTemplateInstance
        #ENDFOR
    #ENDFOR
#ENDFOR
```

The last radio button, Exclude All, doesn't have any #FOR loop at all. So here you will only see the symbols which are in direct view within the template (see the article Understanding the Template Symbols) and only those symbols which you have selected. In this case the template code structure will look something like this:

```
!%%ApplicationTemplateInstance=%ApplicationTemplateInstance
```

After you have made your selection it is just a matter of looking at the source to see the code generated for each template symbol. Before selecting the source button it is a good idea to mark the embed in which you want to use the template symbols. In this way you don't have to search for the embed when entering the source code, because the template-generated symbol values will either precede or follow your embed code, depending on where you have placed the template before or after the embed code.

Now looking through the generated code you will notice that each symbol group has a header looking something like this:

```
!----- Symbols Dependent on %File [1:2] -----!
```

In this case the %File is the selected group name. The [1:2] tells you where in the symbol hierarchy these symbols belong. The first position containing 1 is the same in all cases and represents the application the second position is the %File instance number. In other words this group header has looped through the %File twice. If you move further down the hierarchy more instance numbers will be included. For example, !----- Symbols Dependent on %RelationKeyField [1:4:1:1] -----! tells you, looking the numbers in reverse order, that this is the first instance of the %RelationKeyField, the first instance of the %Relation, the fourth instance of the %File and the first instance of the %Application.

One or two little problems

As already stated this template turned out to be a bit more complicated and time consuming then originally anticipated. It contains nearly all the available template symbols. What do I mean by *nearly* all template symbols? Well, although I haven't been able to make a complete test of the template, I came across a couple of errors, among which one of them I haven't been able to resolve.

The first error I came across was %ActiveTemplateType, which made quite a good job of crashing the Clarion developing environment when I was compiling the module in which the template was used. The Programmer's Guide defines %ActiveTemplateType as:

> The type of all control templates used in the procedure. Multi-valued. Dependent on %Procedure. Returns CODE, EXTENSION or CONTROL.

So what was the problem? In this case it was the term "Multi-valued." For any other template symbols multi-value meant that the symbol could be used within a #FOR loop. So it turned out that the %ActiveTemplateType wasn't multi-valued. The %ActiveTemplateType is dependent on %ActiveTemplate, which is multi-valued. Here all I had to do was remove %ActiveTemplateType from the #FOR loop and include it in the %ActiveTemplate group.

The second error I came across was %FieldDisplayChoices. Ah, I thought, this is the same error as with the %ActiveTemplateType, but no, it was something entirely different. The Programmer's Guide says this:

> %FieldDisplayChoices: The choices the user entered in the display override field for a Must Be In List field (%FieldValidation contains INLIST). Multi-valued.

No matter what I tried I just got an error that stated that the variable is not defined. To make a long story short, I decided to comment out anything to do with the %FieldDisplayChoices in the template. So if you come across a solution for this

template symbol please let me know.

Download the source

Steffen S. Rasmussen has graduated in Computer Science from Copenhagen Business College. Since then he has worked as a programmer, system technician and network administrator, and is currently IT manager. Clarion is a quite a new language to Steffen since his only been working with it since January 2000. But what better way to learn it than by trying to teach others! Steffen has also set up a web site to collect as many examples of different user interfaces as possible to inspire Clarion developers.

Reader Comments

Add a comment

Steffen, This is a very nice template! It really...

A Template For Exporting Classes

by Lee White

Published 2002-09-20

In my previous article about class wrappers I covered the steps required to create a wrapper, but stopping there leaves the picture incomplete and limits the wrapper's usefulness. The most obvious limitation is that a simple class wrapper works fine for single EXE applications, but what about multiple DLL applications?

As you may recall from that article, I had just accomplished writing my first class. The class worked fine, until I decided to plug it into a multiple DLL application I use daily. This proved to be rather devastating - my app belched and died a horrible death accompanied, of course, by an application exception error. Obviously I needed to do more digging.

For a project that includes multiple DLLs and a single EXE you need a means to export the class methods from your data DLL. Without this you're out of luck getting your classes to work across the remaining DLLs and, if needed, the EXE. To this end, I once again began to dig and test, with the obligatory failures, and have arrived at a solution.

In this article I'll cover several different options, including: a global extension where no embedded source or method calls are required (method export only); a global extension where only template generated method calls are required locally (for globally initialized methods); and a global extension where both method calls and embedded source must be accommodated locally.

The requirements of your particular class will determine which option you use in writing your global class wrapper.

Also, as necessitated by the template, you'll find a few tips on designing the class header (your class include file).

The template

Since this template will be used as a global extension it requires the APPLICATION tag in the definition. This makes the template available only at the global level:

```
#EXTENSION(MyClassGlobal,'MyClass Global'),APPLICATION
```

To get things rolling your template needs #PREPARE and #ATSTART sections:

```
#PREPARE
  #CALL(%ReadABCFiles(ABC))
  #CALL(%SetClassDefaults(ABC),'MyClassObj','MyClass',%ClassName)
#ENDPREPARE
```

Ok, what's all this? #CALL(%ReadABCFiles(ABC)) purges a bunch of template symbols which are then reloaded through a #SERVICE call into C55TPLS.DLL. Several of these symbols are required by the actions performed by this template. In addition, since the values are assigned within C55TPLS.DLL, these symbols should be considered off limits - use them but don't mess with them.

For those with curious minds, this call appears to be the trigger that opens the infamous "Reading ABC Header Files…" window, assuming the headers have not already been read.

It took me a bit of effort to grasp the purpose of #CALL(%SetClassDefaults(ABC). The first parameter is the label used when referencing your class methods local to the application in which the template is populated. An example: MyClassObj.MyMethod(). Consider that a global extension, although it can exist multiple times when using the MULTI attribute, has no discernable instance number. In contrast, a procedure extension, code, or control template has its own unique instance that can be referenced using %ActiveTemplateInstance. This provides you with the ability to uniquely name each use of your class using MyClass&%ActiveTemplateInstance. Global extensions don't have that luxury, so you need to specify an instance name. For the scope of this article the assumption is a single instance of the global extension named MyClassObj.

The second parameter is the local object name for your class. Although it can be the same as the first parameter, I prefer using a different moniker for each to help me retain the modest amount of sanity I have remaining. In other words: to reduce confusion.

The last parameter is a template symbol, containing the actual declared class name. This is used later in the template.

For those with knowledge of the template language and its symbols, you may be wondering

why %ApplicationTemplateInstance isn't used to define the difference between multiple instances of a global application template. I considered this myself and ran some tests. The conclusion I reached, at least from my tests, was that although you can loop through the template instances there appears to be no way to associate the instance number with the corresponding template instance. So, it's of little value when defining the label for local use of your template methods. This may simply be an oversight within the template engine or a problem in the version I am currently using: Clarion 5.5.07(g). Whatever the reason, it isn't important here since I'm just dealing with a single instance global extension.

As in the class wrapper, the #ATSTART section is a duplicate of the #PREPARE section. The difference is when these two sections perform their function. #PREPARE is used to *prepare* symbols whenever you access the template, whereas the #ATSTART section is used at the *start* of source generation.

```
#ATSTART
  #CALL(%ReadABCFiles(ABC))
  #CALL(%SetClassDefaults(ABC),'MyClassObj','MyClass',%ClassName)
#ENDAT
```

#INSERT(%OOPPrompts(ABC)) includes a group of hidden prompts already defined by the ABC template chain in ABGROUP.TPW. These hidden prompts are place keepers for symbols used in the remainder of the template so it is required – don't leave it out!

Next comes the more obvious and common section, the #SHEET.

```
#SHEET
  #TAB('Setup')
    #DISPLAY('Your global template prompts')
  #ENDTAB
  #TAB('Class')
    #PROMPT('Class:',FROM(%pClassName)),%ClassName,DEFAULT('cMyClass'),REQ
    #BOXED('')
      #BUTTON('My Class'),AT(,,170)
        #WITH(%ClassItem,'MyClassObj')
         #INSERT(%GlobalClassPrompts(ABC))
        #ENDWITH
      #ENDBUTTON
    #ENDBOXED
    #BUTTON('Library Files'),AT(,,170)
      #BOXED('Library Files')
        #INSERT(%AbcLibraryPrompts(ABC))
      #ENDBOXED
    #ENDBUTTON
  #ENDTAB
#ENDSHEET
```

Depending on the requirements of your class, when used globally, you may or may not have a need for any template prompts other than the requisite prompts displayed on the second #TAB

as shown above.

The first prompt on the second tab provides user access to the %ClassName symbol used in the #PREPARE and #ATSTART sections. %pClassName is one of the many symbols loaded by the #SERVICE call mentioned earlier, and is a multi-value symbol containing a list of all known classes. The default value for this prompt is your class name. This tab also includes template source for the standard global class prompts.

But first, a class header tip

Before continuing I need to cover a few concepts of the class headers. To begin, the header is your class include file: MyClass.inc, for example.

One of the often aggravating aspects of writing your classes is that if you add them to the libsrc directory they are included in every application you compile. That is, unless you use a bit of information I garnered from a well-known source, Russ Eggen. Russ recommended that I change a couple of things in my first class. These suggestions pushed me to discover their use as well as their usefulness, all of which became ever more significant as I worked on this template.

If you have taken the time to poke around in the base class include files you have, no doubt, noticed that odd recurring line found in all of them: !ABCIncludeFile. This is a required statement if you want the IDE to load your classes during the "Reading ABC Header Files…" process.

You may have also noticed that in some cases a parameter is added to this statement, such as: !ABCIncludeFile(MYCLASSINC). This parameter is very important, as is the fact that it's uppercase. Internally the IDE maintains this value as uppercase and its usage elsewhere is case sensitive. Call it what you may: a feature, a limitation, or whatever. This parameter *must* be uppercase – don't forget this!

Regardless of this slight nit, the added benefit of this parameter is that it restricts the inclusion of classes to only those applications that explicitly reference those classes. This alone makes it worth the effort since it can reduce the size of your compiled applications.

Also important are the LINK() and DLL() attributes. These attributes determine the behavior of the class dependent on application project settings.

```
cMyClass  CLASS,TYPE,MODULE('myclass.clw'),↵
   LINK('myclass.clw',_ABCLinkMode_),DLL(_ABCDLLMode_)
```

The LINK() attribute instructs the compiler to include the indicated file, named in the first

parameter, in the link list. The value of the second parameter, a #PRAGMA project define, determines whether the file is included. If this second parameter is present in the project, and has a non-zero value, the file will be included. The file will not be included if the parameter is undefined or has a value of zero.

The DLL() attribute specifies whether the class is defined locally or in another DLL. If its single parameter, another #PRAGMA project define, is undefined in the project, or has a value other than zero, it indicates the class is exported from another DLL.

Pay close attention to the text that precedes the LinkMode_ and DLLMode_ parameters. Disregarding the leading underscore the prefix for these parameters can use ABC, as in the base classes, or can be a more obvious reference to your class using something similar to MyClass. Personally, I prefer to associate the parameters with my class. Either will work but whichever you choose must be referenced correctly in the template sections below. For this article I've used the reference approach: MyClass. Wonder how MyClass gets assigned a value? I'm getting to that.

If you decide to associate the parameters with your class, as I do, don't worry about any additional requirements in the templates – the AppGen handles it for you by adding your parameters to the project. Since it is handled automatically, I define parameters associated with my class making it easier to locate and determine their respective values when exported to a TXA file.

Bottom line, it's your choice.

Decisions, decisions...

I've covered the basic template sections as well as a few bits and pieces of the class header file; now come the optional sections where your particular needs will determine what is used and what isn't.

If you simply need to export your class methods, and absolutely nothing else, you need only add this one additional section to your template and you're finished:

```
#AT(%BeforeGenerateApplication)
  #CALL(%AddCategory(ABC),'MYCLASSINC'[1])
  #CALL(%SetCategoryLocation(ABC),'MYCLASSINC'[1],'MyClass'[2])
#ENDAT
```

In both calls you should notice the use of the uppercased 'MYCLASSINC' parameter [[1]]. This is the reference to the !ABCIncludeFile() parameter as mentioned above. Remember, it *must* be uppercase to function properly.

The second parameter [2] of `#CALL(%SetCategoryLocation(ABC)` is the prefix for the `LinkMode_` and `DLLMode_` parameters of your class definition. Although not case sensitive, the prefix should follow the same form as in your class definition for easier recognition. This prefix is inserted between the leading underscore and the remaining parameter text. So, assuming you use `'MyClass'` as the second parameter in `#CALL(%SetCategoryLocation(ABC)` the AppGen will add these lines to the internal application project before compiling:

```
#pragma define(_MyClassDllMode_=>0)
#pragma define(_MyClassLinkMode_=>1)
```

The actual values will depend on whether this is a DLL or an EXE. One note of interest: as I mentioned, the prefix is not case sensitive but there is one caveat to its use. If you change the case of the prefix, at some later time, you will end up with multiple `#PRAGMA` defines in your project, for each different use of case, with the potential to cause problems with incorrect values. So be certain about the case of your prefix before you use it the first time.

For class method export only, you're done! If your requirements include global use of your methods or access to embedded source, within the current application, there are a few other template sections needed.

Global method calls

Before your template writes anything, it must gather all applicable information. The `%GatherObjects` embed provides this functionality. In this case all you need to do is add you global object name, as described earlier, to the application object list.

```
#AT(%GatherObjects)
  #CALL(%AddObjectList(ABC),'MyClassObj')
#ENDAT
```

Adding your object to this list will insure its inclusion into the application global list of objects so it can be referenced using `MyClassObj.MyMethod` syntax.

The next segment makes the call that writes the definition for your class into the application.

```
#AT(%GlobalData)
#INSERT(%GenerateClass(ABC),'MyClassObj')
#ENDAT
```

For those occasions where you only need to include calls to your methods, without embed support, you can now add the `#AT()` calls you need using syntax similar to these examples:

```
#AT(%ProgramSetup)
%ThisObjectName.Init()
```

```
#ENDAT
#AT(%ProgramEnd)
%ThisObjectName.Kill()
#ENDAT
```

If you have no need for embed support, once your #AT() calls are written, you're finished. On the other hand, if you do need to provide embed support, keep going!

Global embeds

Providing embed access begins with a call to %GenerateVirtuals.

```
#AT(%ProgramProcedures)
#CALL(%GenerateVirtuals(ABC),'MyClassObj',↵
   'Global Objects|MyClass Description',↵
   '%GlobalEmbedVirtuals(MyTemplateSet)',%TRUE)
#ENDAT
```

%GenerateVirtuals is a group defined in ABGROUP.TPW. This group takes four parameters and handles embed generation. The parameters, in order, are: the class object name, your embed tree description, the name of a locally defined template group, and a Boolean. This last parameter, the Boolean, changes the embed prototype for new methods derived from your base class by eliminating (when true) or including (when false) the use of %ActiveTemplateInstance. Your global template does not have multiple instances that can be referenced using this symbol so the example above passes a true value to eliminate its use.

The following section is responsible for generating the PARENT.Method calls within each embed generated by the prior call to %GenerateVirtuals.

```
#AT(%MyClassMethodCodeSection),PRIORITY(5000),↵
   DESCRIPTION('Parent Call'),WHERE(%ParentCallValid())
#CALL(%GenerateParentCall(ABC))
#ENDAT
```

The last two sections are the only local groups needed by the wrapper if you require embed access to your methods:

```
#GROUP(%GlobalEmbedVirtuals,%TreeText,%DataText,%CodeText)
#EMBED(%MyClassMethodDataSection, ↵
   'MyClass Description Method Data Section'),%pClassMethod,↵
   %pClassMethodPrototype,LABEL,DATA,TREE(%TreeText&%DataText)
  #?CODE
  #EMBED(%MyClassMethodCodeSection, ↵
   'MyClass Description Method Code Section'),%pClassMethod,↵
   %pClassMethodPrototype,TREE(%TreeText&%CodeText)
```

Those four lines define the group called by %GenerateVirtuals with %GlobalEmbedVirtuals(MyTemplateSet). This is an important piece of the puzzle since this group generates embeds for your methods.

And finally, this is the group called by the section that generates the PARENT.Method calls:

```
#GROUP(%ParentCallValid),AUTO
#DECLARE(%RVal)
#CALL(%ParentCallValid(ABC)),%RVal
#RETURN(%RVal)
```

By using a local group that can be called directly from a WHERE() clause you alleviate several other groups otherwise required in your wrapper. Since WHERE() in this instance cannot include the required (set) parameter to call a group that resides in a separate template set, such as ABCHAIN.TPL, your only alternative is to duplicate this external group, and all subsequent groups it calls, or use a local group with #CALL since it can use this parameter.

Wrapping it up

Please don't be misled; this is not the 'do all' of global class wrappers. More than likely you will need to omit some template calls, as well as embed code, using a combination of application symbols such as %GlobalExternal and %ProgramExtension.

Due to the number of possible combinations and requirements, I have purposefully resisted including either of these symbols in this article for fear they might, indeed, be misleading. Remember, %GlobalExternal is true when global data is external to the current application, regardless of program extension, and %ProgramExtension is the extension you indicated in your application project: EXE, DLL or LIB.

Consider the use of your template in a global data DLL where your file schemas, global variables and class methods are exported. Generally no other functionality is included here so calling methods from your class or allowing embed support would be superfluous. On the other hand, when you use this template in an application that generates an executable, it would make sense to call any necessary methods and possibly include embed access. You might consider using something similar to this:

```
#IF(%GlobalExternal OR %ProgramExtension='EXE')
  <accessible except in data DLL>
#ENDIF
```

Deciphered: if this is not a data DLL or it is an EXE, grant access.

For restricted use of #EMBED statements you could use the logic of the #IF above in the statements' WHERE() clause, such as:

```
#EMBED(%MyEmbed,...),WHERE(%GlobalExternal OR %ProgramExtension='EXE')
```

In the simplest of multi-DLL cases you can export your class methods from your data DLL using a global extension, and then use your procedure extension, code or control templates to access those classes without the need for any additional global extensions. If you have a single EXE and you don't need global method calls, a global extension is unnecessary.

I hope this gives you a good start on your global class wrapper. Enjoy!

Download the source

Lee White's introduction to computer languages began during his sophomore year in high school, through an open class at the University of Alabama, c1969. His first language was Fortran 4 with Watfor. Introduced to Clarion 2.1 in 1990, he has remained a steadfast Clarion user and supporter. He is often better known for his contributions to the etc conferences and third party products.

Reader Comments

Add a comment

Clarion Magazine

Data Structures And Algorithms Part IX - Are You Getting Too Tall?

by Alison Neal

Published 2002-09-26

In my last article I discussed Trees, and more specifically the Weight Balanced or Perfectly Balanced Tree. The Weight Balanced Tree is one of the approaches used to overcome the shortcomings of the Binary Search Tree. Another alternative is the Height Balanced Tree, or AVL Tree.

The Height Balanced Tree ensures that neither of the sub trees (left or right) gets too tall. If a Binary Tree gets overly tall then searching can get very slow. As with the standard Binary Search Tree, if the data is entered in sorted order, the structure collapses into a linked list, and every time an action is made on the tree, you have to start at the head node and visit every subsequent node. This defeats the idea of the Binary Tree, which is to hopefully halve the processing time.

Height Balanced Binary Search Tree

On first look, the Height Balanced Binary Search Tree may not always appear to be balanced. Figure 1 shows a height balanced tree that looks a bit lopsided.

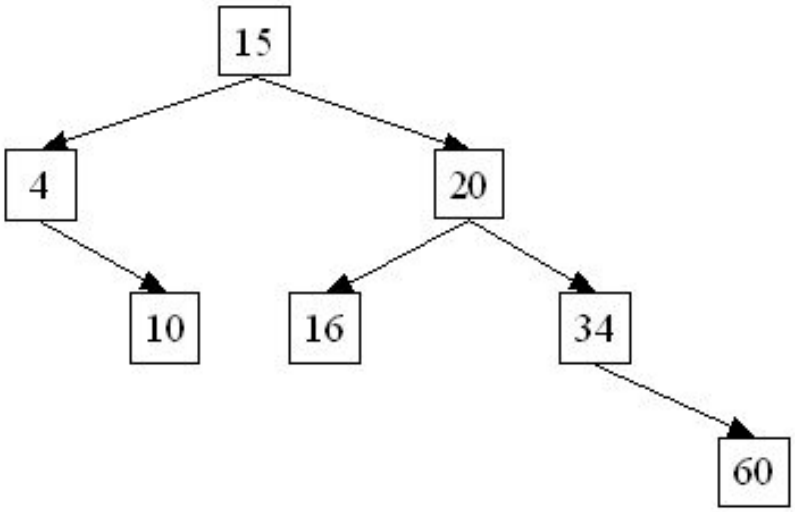


Figure 1

Looking at Figure 1 you should be able to see that neither left or right sub trees varies in height by more than one. The height of the left sub tree is two and the height of the right sub tree is three. There are only two levels in the left sub tree and three levels in the right sub tree. The *weight* of the tree is not balanced, however as the left sub tree has a weight of two and the right sub tree has a weight of four. AVL trees or height-balanced trees, however, do not consider weight only the height. One side is not more than one level taller than the other, so by AVL standards this tree is balanced.

The methods I've used in my implementation of the Height Balanced Tree are as follows:

Method	**Description**
`Init`	Initialise the root of the tree
`Kill`	Dispose of the tree
`IsEmpty`	Check to see whether the tree is empty
`Insert`	Add a data item to the tree
`Height`	Get the height of the tree
`Find`	Find an element in the tree
`InOrder`	Traverse the tree in order
`MorrisInOrder`	Traverse the tree iteratively in order

Output	Print the tree

The init, Kill, isEmpty, Height and Find methods are exactly the same for the Height Balanced Tree as they are for the normal Binary Search Tree.

The insert method, however, was a lot of fun to write. It requires me to maintain the height of the tree, to do this I've added a "balance" variable to the structure:

```
treeNode    CLASS,TYPE
NodeVal     ULONG(0)
Balance     SHORT(0)
ltree       &treeNode
rtree       &treeNode
            END
```

The balance variable allows me to keep track of how many nodes have been added to the left and right sub trees. For example, when a node is originally created the balance is automatically set to zero, if I then add a node to the left sub tree of the original, then the original node's balance has 1 subtracted from it, so that the balance equals –1. If I then add another node to the right sub tree 1 is added to the original nodes balance, so the balance goes back to equalling zero, which means that the heights of both sub trees are balanced.

However, if I then add two more levels to the left sub tree, the balance will become 2. This is the point (either 2 or –2) where I know for certain that the tree has become out of balance and so the nodes need to be rotated around accordingly.

I'll now walk through the insert methods with an example, and hopefully clarify the logic further:

```
AVLTree.Insert           PROCEDURE(ULONG yourVal)
done    BOOL(FALSE)
    CODE
    SELF.root &= SELF.AVLinsert(SELF.root,yourVal,done)
AVLTree.AVLinsert PROCEDURE(*treeNode t, ULONG yourVal,*BOOL done)
temp    BOOL
    CODE
    done = FALSE
    IF t &= NULL
      SELF.Curr &= NEW(treeNode)
?     ASSERT(~SELF.curr &= NULL)
      t &= SELF.curr
      t.NodeVal = yourVal
      t.ltree &= NULL
      t.rtree &= NULL
      t.balance = 0
```

```
      done = TRUE
    END
    IF (yourVal > t.nodeVal)
      temp = done
      t.rtree &= SELF.AVLinsert(t.rtree,yourVal,done)

      IF done
        done = temp
        t.balance += 1
        IF t.balance = 1  THEN done = TRUE.
        IF t.balance = 2
         IF t.rtree.balance = -1 THEN t.rtree &= SELF.RR(t.rtree).
          t &= SELF.LR(t)
        END
      END
    ELSIF (yourVal < t.nodeVal)
      temp = done
      t.ltree &= SELF.AVLinsert(t.ltree,yourVal,done)

      IF done
        done = temp
        t.balance -= 1
        IF t.balance = -1 THEN done = TRUE.
        IF t.balance = -2
          IF t.ltree.balance = 1 THEN t.ltree &= SELF.LR(t.ltree).
          t &= SELF.RR(t)
        END
      END
    END
    RETURN t

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

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

Note that there are four insert methods in all: the first calling method, the recursive insert method, `LR` (Left Rotate) and `RR` (Right Rotate).

Take a look at what happens when I try to insert the following numbers:

37	26	5	21	20	47	48	35	43	6	32

The first call to the `Insert` method occurs with value 37 to be inserted into the tree. The `Insert` method calls `AVLInsert`; since the root of the tree is null the ancestor node is allocated and 37 is allocated. Figure 2 shows the current tree with a zero balance.

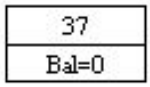


Figure 2

Now the value 26 is added, as it is lower than 37, a node is created in the left sub tree with a zero balance. On returning to the head node, the balance has 1 subtracted, so now my tree looks like figure 3.

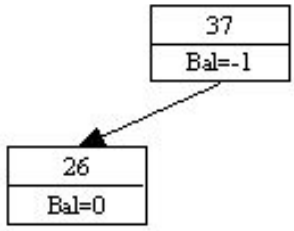


Figure 3

Next the value 5 is added. This is lower than 37, so a recursive call is made to the left node. 5 is also lower than 26 so another recursive call is made to the left. At this point a new node is created and 5 is assigned. On returning to 26, the balance is adjusted accordingly to –1, and on returning to 37 the balance is adjusted accordingly, only this time the balance now equals –2. As I stated above, when the balance equals either –2 or 2 I can be certain that the tree is no longer height balanced, and therefore I need to rotate the nodes accordingly. Before the rotate method is called the tree looks like Figure 4.

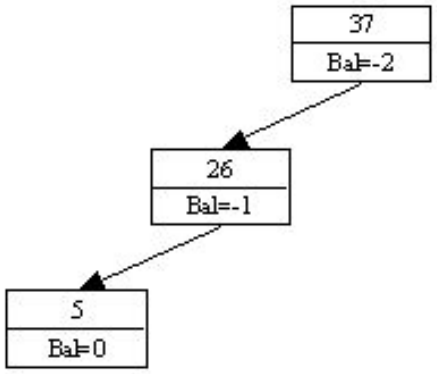


Figure 4

What needs to happen now that the tree is evidently out of balance? The tree needs to go through a Right Rotation, and so a call is made to the `RR` method.

The first thing to happen in the `RR` method is that `Q` is initialised to refer to the root node (`37 -> 26 -> 5`), and `t` is made to refer to the left sub tree (`26 -> 5`). `Q`'s left sub tree is then made to equal null, as `t`'s right sub tree is currently null. So now `Q` is just 37, as it no longer refers to any other nodes. `T`'s right sub tree is then made to refer to `Q`. So, effectively 26 is now the root node, 5 is in the left sub tree, and 37 is in the right sub tree. The balance of `Q` is altered so that 37 now has a balance of –1 (rather than –2).

As t.balance is –1 (26), `q.balance` is then made to equal zero. This makes the balance of `q` (37) correct because in its current state `q` has no sub trees. However, `t` (26) still has a balance of –1, which is incorrect so 1 is added to make this balance now zero. Figure 5 shows how the tree looks on exiting the `RR` method:

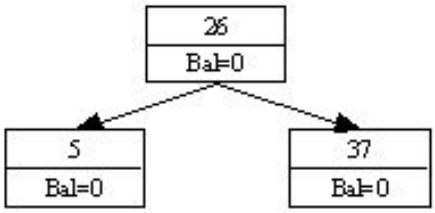


Figure 5

Note that the balance of the new root node is zero. This is not because it doesn't think there are any nodes in the sub trees, but because the height of the sub trees are even. The height difference between the left and right sub trees of a height-balanced tree should only ever differ by at most one.

The next value added is 21, which is less than 26 but greater than 5, and so gets allocated to the right sub tree of 5, resulting in Figure 6:

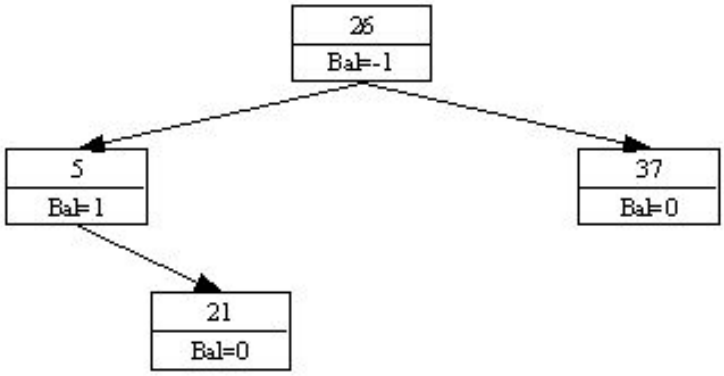


Figure 6

The next number to be added is 20, which is less than 26, greater than 5, and less than 21. So a new node is allocated in the left sub tree of 21. This time on returning to the node with the value 5, its balance is incremented to 2, which indicates that the tree is out of balance, as per Figure 7:

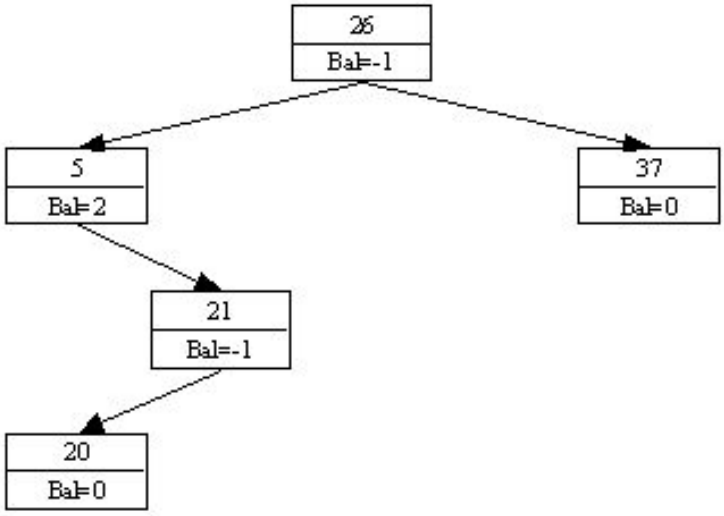


Figure 7

However, unlike last time, where 36, 27, 5 were nicely in line to the left, the nodes are now crooked, so a pure left rotate at this point wont work. So, first I have to right rotate 20 and 21.

The `insert` method knows that the numbers are not in a row because the balance of 21 is currently –1, and it calls the `RR` method passing the right sub tree of 21 and 20.

So, this time q is made to reference 21 and t is made to reference 20. Q's left tree is made to equal null, as t's right sub tree has nothing assigned. T's right sub tree is then made to equal 21. So now 20 points down to 21, in its right sub tree rather than the other way around, and the nodes are now in a straight line leading off to the right. Q's balance is made to be zero, which is correct as at this stage it doesn't have any sub nodes and t's balance is incremented by 1 indicating that it now has a right sub tree. So now the tree looks like figure 8:

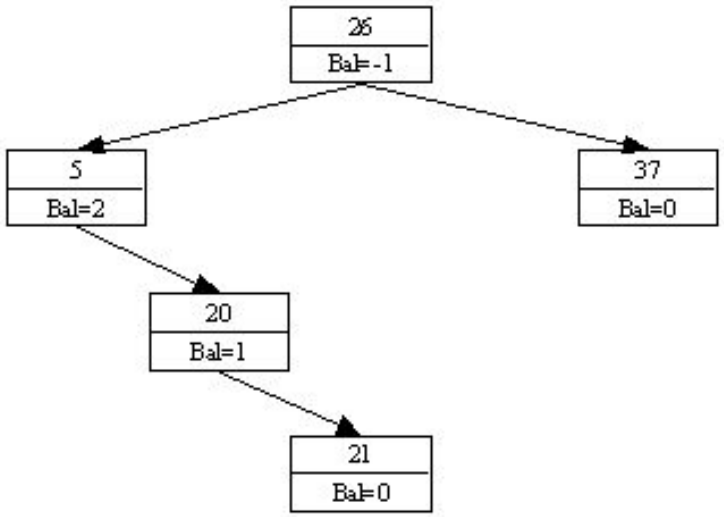


Figure 8

Now the Left Rotate (LR method) can be performed for the node containing value 5. This is exactly like the Right Rotate except that it works in the opposite direction.

Q refers to the node containing value 5 and t is made to refer to the node containing 20. As t (20) has nothing in its left sub tree, q's (5) right sub tree is effectively made to equal null. This means that it no longer refers to any of the lower nodes. Q is then assigned to t's left sub tree and the balances are altered accordingly with q's balance becoming zero and t's balance becoming zero. Figure 9 shows the new tree:

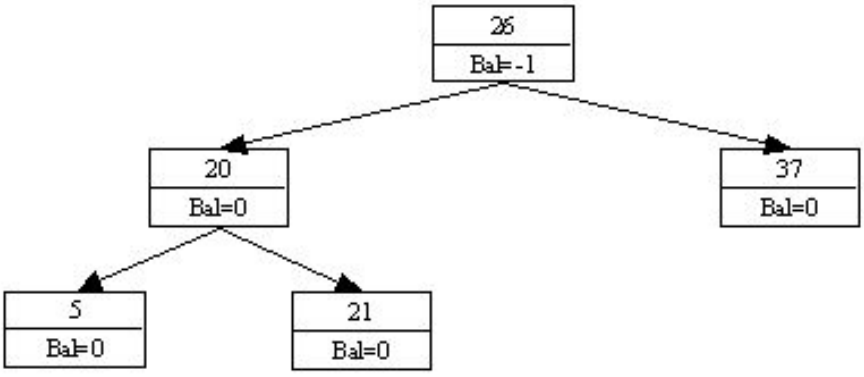


Figure 9

On returning to the head node "done" is not true, so the root node does not have its balance decremented again. Note that "done" is a reference variable that is set to FALSE at the beginning of the recursive function, and is only set to true when a) a new node is allocated, thus triggering the checks for changes in balance for the node immediately above the new node and b) when the height of a child node differs by one and therefore the parent nodes balance may also be incorrect and needs checking.

Just before the process checks to see if the nodes are out of balance the "done" variable is reset to it's original FALSE value (stored in `temp`). This means that the balance variable from the parent nodes will only be updated when rotations have not already occurred among the child nodes or when a new level has been added to the tree.

The next number to be added is 47, this is greater than 26 and 37 and so slots nicely into a new right-most node. All the balances are updated accordingly (Figure 10):

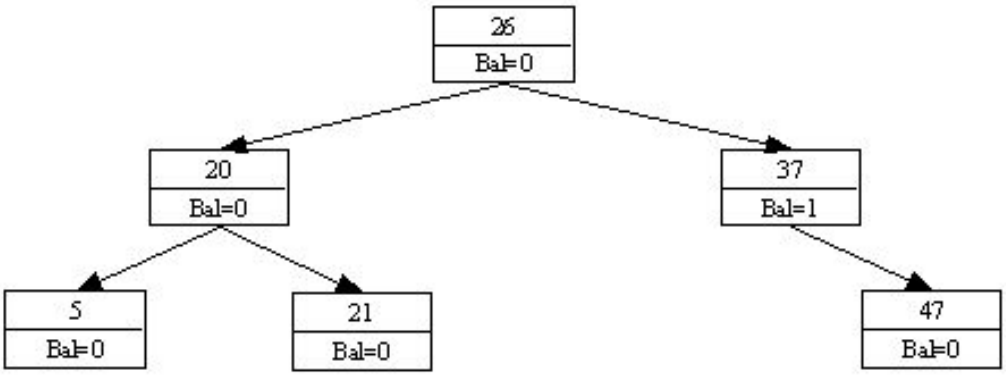


Figure 10

The next number to be added is 48, which instigates the creation of a new level in the right sub tree, causing 37 to be out of balance:

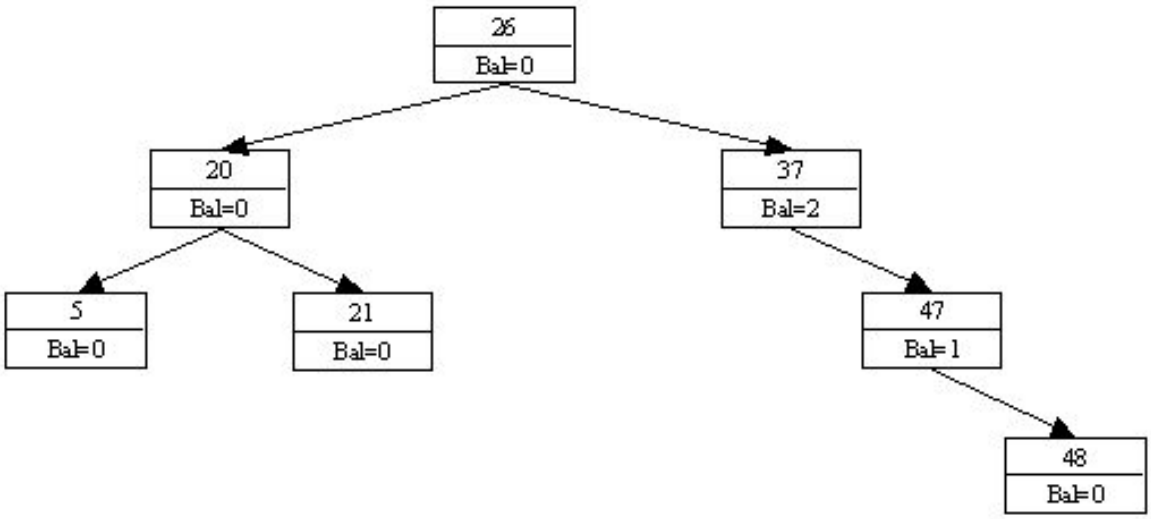


Figure 11

Now a left rotation around 47 must occur for the tree to come back into balance:

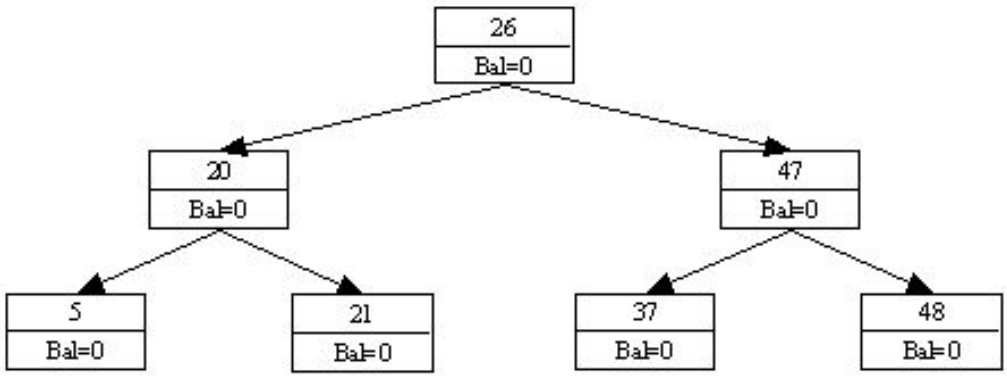


Figure 12

At this stage in the process, my tree is not just height balanced but it is also perfectly balanced. Unfortunately I now want to add 35 and the result is shown in Figure 13:

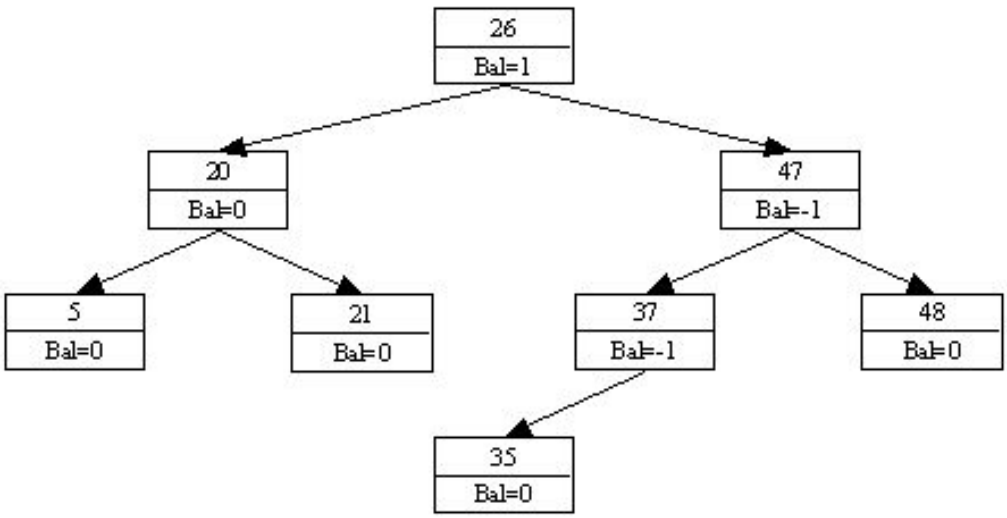


Figure 13

The left and right heights in Figure 13 are within one, so the tree is still considered balanced. Next I insert 43, giving Figure 14.

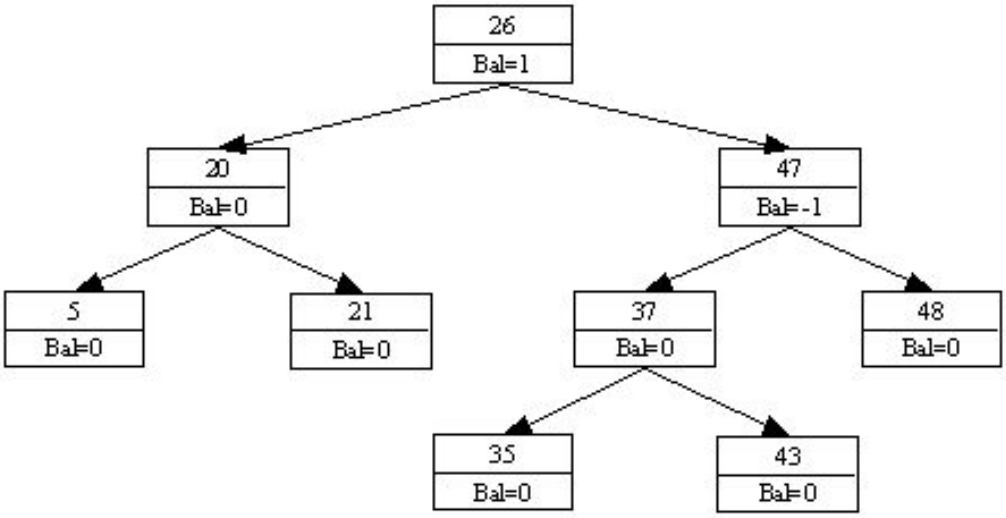


Figure 14

The insert of value 43, still hasn't caused the creation of a new level,so the tree is still considered height balanced. The heights of the left and right sub trees still do not differ by more than one.

The value 6 is then added giving Figure 15:

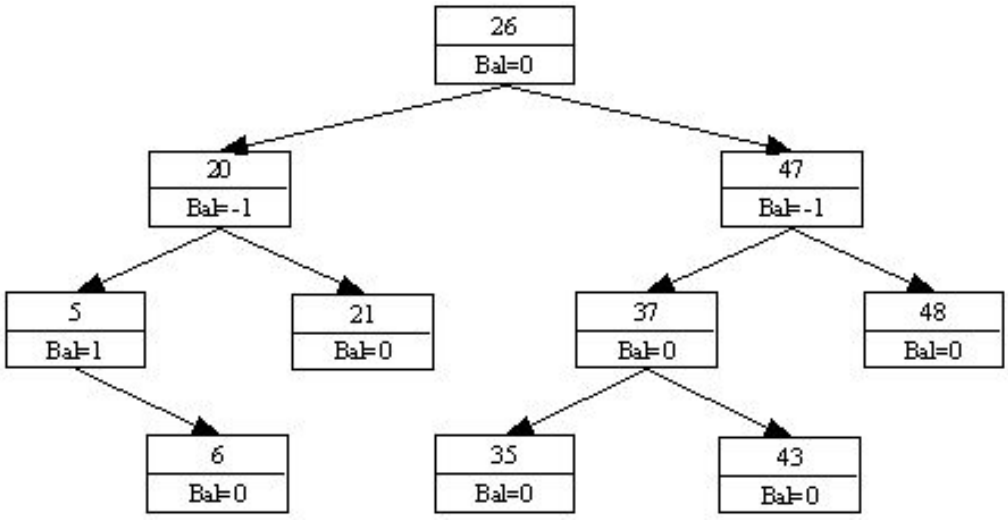


Figure 15

Then the last number 32 is added. 32 is larger 26, and less than 47, 37 and 35, so when the new node has been added, but before the rotations are called, the tree looks like Figure 16:

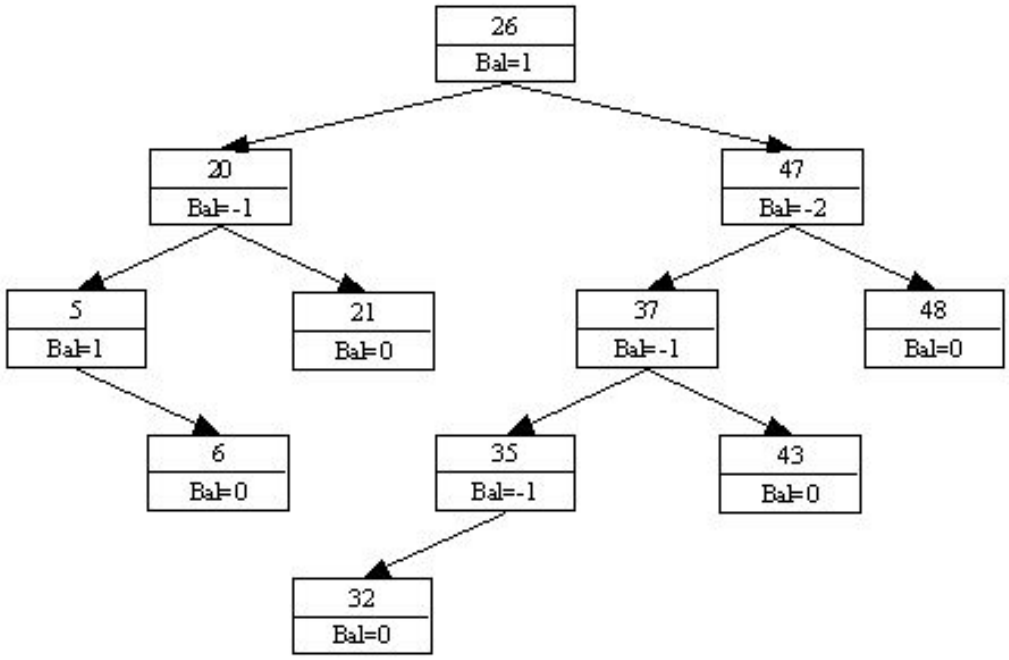


Figure 16

From Figure 16 it can be seen that the node containing 47 has sub trees that are now out of balance by more than one, so a right rotation needs to occur.

When the RR method is called, `Q` is made to refer to the node containing 47, and `t` is made to refer to the node containing value 37. `Q`'s left sub tree is then made refer to `t`'s right sub tree. So now 47 has a left sub tree of 43 and a right sub tree of 48. `T`'s right sub tree is then made to refer to the node containing 47 and the balances are updated accordingly, giving:

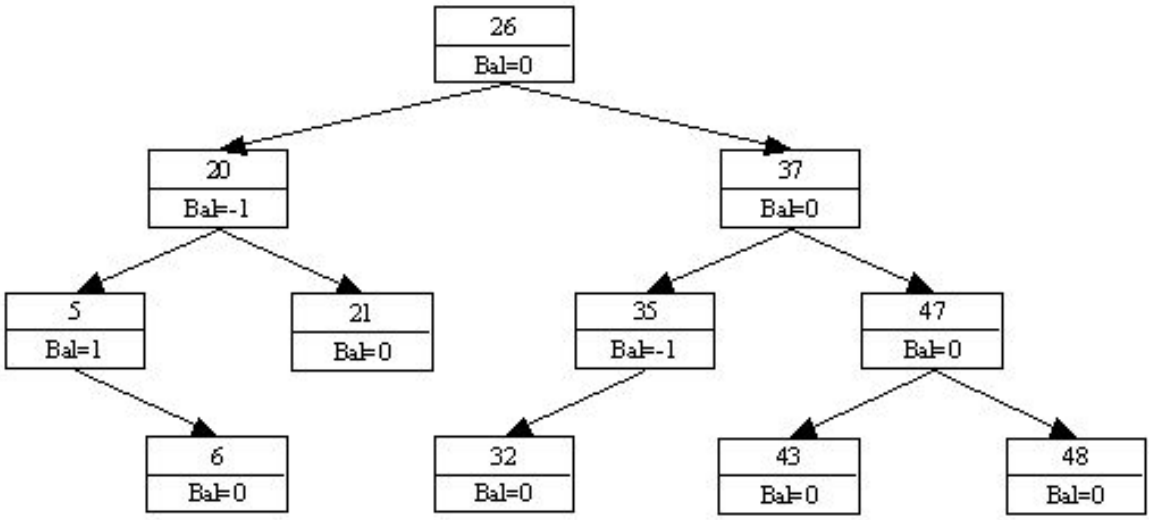


Figure 17

Summary

Height balanced trees are a lot of fun to play with, and are a good option when multiple searches are going to be applied. However, as you can see, there are large overheads when inserting items, so they are not always a good option where frequent inserts and changes are required.

Deleting individual items from an AVL tree is also quite difficult, so I will cover this in my next article along with Tree traversal. There are three generally accepted traversal techniques used with the Tree data structure, these include in-order, post-order and pre-order; also the tree does not always have to be traversed using recursion, it can be traversed using iteration, using the Morris In-Order traversal algorithm. More on all that next time.

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

Reader Comments

Add a comment

http://www.ucomics.com/foxtrot/2002/09/27/

Topics

Advertising Feature: XP Menu - A Simple And Extendable Clarion User Interface

by Ronald Raaphorst

Published 2002-09-26

> *The following advertising feature about XP Menu, a commercial third party product, is written by the author of that product. It is not a review, and does not constitute part of anyone's paid subscription, nor is it an endorsement of this product by Clarion Magazine. Clarion third party vendors wishing to publish advertising features should contact dharms@clarionmag.com.*

In this article I describe a user interface, which can be (and in part already is) implemented in the Clarion templates and source. This UI is consistent and extendable, both for the programmer and the end user. It uses a side menu, like an XP Menu, which replaces almost all user input action controls like buttons and tabs on a window, leaving more space for data. One obvious advantage for this is functionality can be automatically focus driven; the developer can disable all items that are unavailable, yet the end-user can still see all of the functionality provided in one place. The design solution I present here is template driven, and templates can be written to extend a window's functionality using the XP Menu for controlling and accessing that functionality. The XP Menu is a commercial third party tool, available on www.compad-software.com/uk/developer. This article will be about writing templates for extending functionality using the XP Menu.

Introduction

Clarion is a powerful tool for building database applications. But in all the Windows versions, not that much has changed by way of user interface (UI) for the ABC and Clarion templates. Basically, a browse is a browse, and a form is a form. You can browse through a list, and edit and print data.

The standard Clarion UI is, in my opinion, not good enough for current users and not flexible enough for current programmers. Even though Clarion offers a way of extending functionality by changing templates, or writing additional templates for third party products, I doubt if this is done by the majority of Clarion developers. Mostly, I suspect, Clarion developers enter code copied that code into other embeds, over and over again. A lot of third party vendors offer products to spice up an application. But merely changing a background and adding icons will not make a program better. More controls, more icons and scattered, different sized controls will only draw the users attention away from the data

Let's face the truth: most Clarion programmers (myself included) are not UI specialists nor graphic designers. We do not have the money nor the time to do user group tests, keep up to date with new UI principles, or design icons. But most programmers can do excellent analysis, write a dictionary, create some functional windows, and sell the program. So what's the big deal?

Support

Most of my clients are happy at first just using a program. Instead of reading a manual, they just start and see where they end. The more difficult and the less consistent a UI is for an end user, the more problems the end user will experience, and the more he/she will call/mail you.

My users do not read manuals, and they get confused if I tell them there are several ways to access some particular functionality. So, a program should state clearly what functionality is provided and what functionality is temporarily disabled. And it would be nice if there's just one place to look.

Maintenance

From a maintenance perspective, the more difficult the UI , the messier it gets when you make changes later. If you already have a lot of buttons on a window, and you need another one, where do you place it? Is there enough room in the window?

As said earlier, a UI should be consistent. But it takes some programming discipline to re-create every UI piece in the same way for every window, especially if you add functionality. Wouldn't it be nice to just have to worry about the functionality itself, and use one standard way to provide access?

Getting the picture

Take for example a browse box. A browse box simply displays records. Being able to update a record requires three buttons: Insert, Change and Delete. If you add tagging, you will need at least another button: Tag/Untag. You may want to add Tag All and Untag All buttons also.

Printing records requires another button. That makes six buttons, all of which focus on that single browse box.

Now imagine a child browse box in the same window. Now there are 12 different buttons, with duplicated functionality, focusing on different data sets.

To cut a long story short, here's what I think we really need:

- One control for one user action, which can be used for different targets (Insert can be used for different browses, depending on which browse has the focus).
- A disabled or hidden state for a control under certain conditions, focus driven (i.e. the user can't change a record if there is no record)
- The ability to visually group functionality. Doing something with a record (insert, print) is something different than selecting another tab for displaying other data.
- The ability to easily extend functionality without having to redesign a window.

A solution

While thinking about these problems, I saw the first Windows XP Screenshots. And there was the solution: an XP Menu. As opposed to an Outlook menu, which takes too much space and hides most of its functionality, an XP Menu is compact, and still flexible. An XP menu is always visible, and context sensitive, for you can hide menus or disable menu items when they are not applicable. Still, a user can pretty much see most of what it has to offer. You can group functionality, and an end-user can control what is visible.
The idea is that every window has it's own XP Menu. The menu can mimic controls (like buttons and tabs). The mimicked controls are hidden and accessed by clicking an item in the menu. Actually, the XP Menu post an event (Event:Accepted in most cases) to the (hidden) control. This behaviour is more or less copied from the PopUpClass. The XP menu can also post events based on a condition, or only if a control is visible. Menu's can be hidden automatically, based on a condition. Items can conditionally be enabled or hidden, and conditionally setting an icon for a menu item is also possible.
In future updates, an XP Menu might be extended to control other UI elements in other procedure threads. (I'm thinking about that).

Take a look at the interface in Figure 1. This window contains 2 browse boxes, where the lower is a parent of the upper browse box. The browse box with focus is blue, and the browsebox without focus is colored gray, just like the MS Outlook.

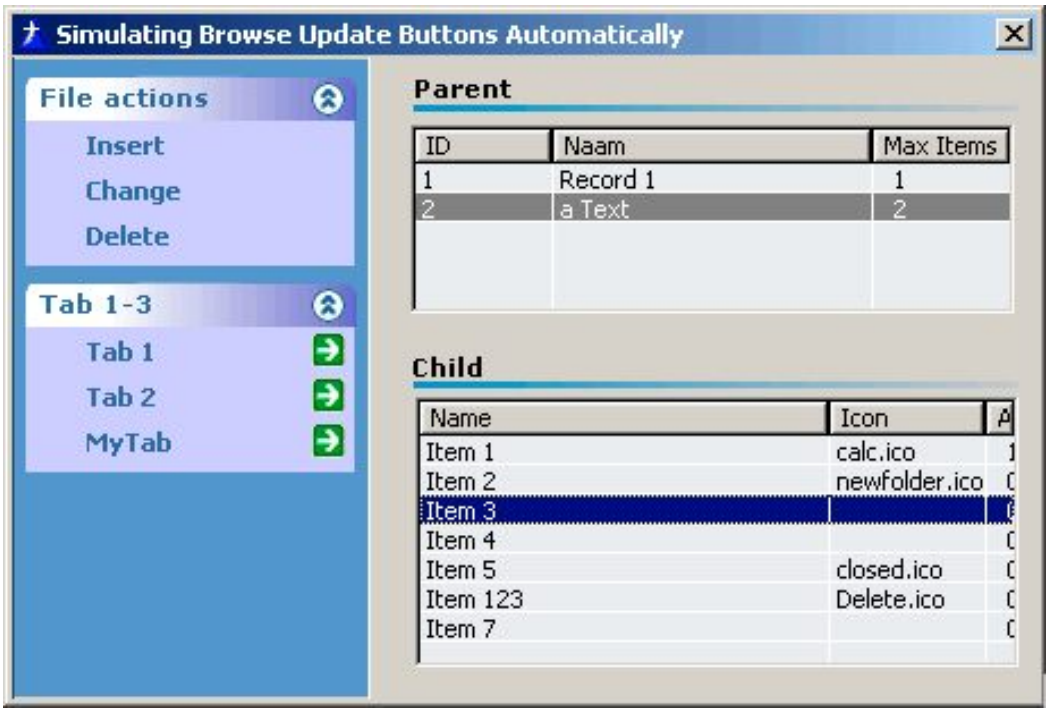


Figure 1. An XP-style menu with two browses

Both browse boxes have hidden BrowseUpdateButtons. If a user clicks an item of the File actions menu, the menu automatically mimics the update button of the browse box with focus. If the user clicks on a record in the parent browse box, it will automatically get focus, turn blue, and will receive future file action clicks from the menu.

If you want to extend the functionality, you just add another item to the file action menu, and that item can be also focus dependent. No buttons, no extra resize strategies, and no messy window. And the end user will search the functionality on the only spot he/she knows: in the menu.

The Tab 1-3 menu is an automatically generated menu, which simulates tabs on a given sheet. Especially if you have a lot of tabs (say, a large update form), it can be nice to have them all in a single list, instead of having multiple tab rows (or use scroll buttons). The names presented in the tab menu are taken from the text on the tabs (?Tab3 is named ‘MyTab’), and the for the sheet, PROP: NoSheet and PROP:Wizard are set. More appropriate names are required though.

Browse functionality

This example shows that the menu is not merely a replacement for buttons and tabs on a window. Even user-controlled selection lists can be implemented in the menu, using the menu to easily control complex behaviour.

For example, I recently found myself creating keys and locators for new fields in the database, in all the windows where the file is used in a browse box. Really, I wanted the end user to be

able to choose which fields should be displayed. This way, if I add another field, it's initially not displayed, but at least the end-user can choose to display it at run-time, and I do not have to test every browse box at compile time. I call this *field selection*.

Also, an end user should be able to choose a sort order from the fields that are displayed in the browse box. Multiple fields can be chosen, just like a key has multiple components, and it should be possible to also select fields for which no key is defined (that may take a little longer at run-time to sort, though). I call this functionality *sort selection*.

There's one more thing: If the same type of data is used in different browse boxes, the same set of field and sort selections can be applied. If for example, I have a browse with orders, I can use this browse box in a BrowseOrder window, and in a BrowseClient or UpdateClient window. In the first window, all orders are presented, and in the additional windows, only those orders belonging to that client are presented. So there is a browse box with orders, and in some cases, the order list may be range limited while displaying the same columns. Technically, the fields and sort settings should be usable in different browse boxes when the same view is used.

Field and sort selection – visual implementation

The window in Figure 2 updates a Dossier record. (Dossier is a table in a patent registration application I created) The code and title fields are actually fields from this table, and the list box (on the invisible tab 'Contract') shows contract child records, which belong to the dossier. This window contains a lot of tabs, but the Extra menu here is set to show a maximum of six items (that's changeable by the end-user at run-time per menu though).

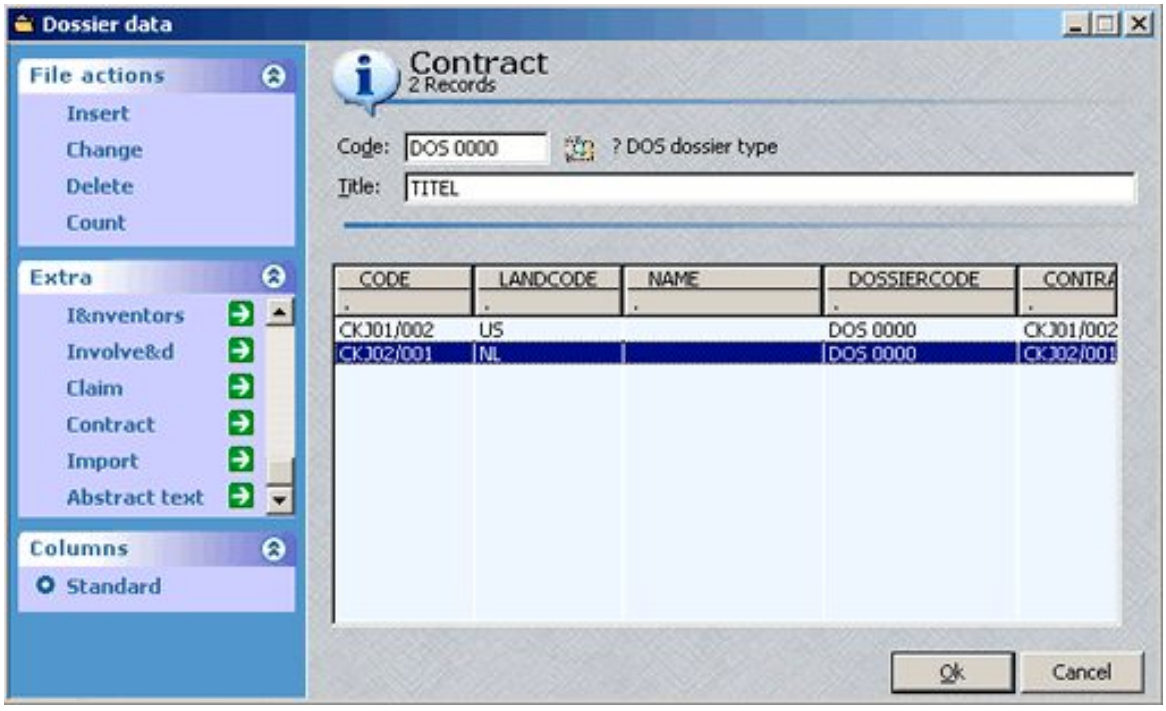


Figure 2. The Contract "tab" on the Dossier update form.

Note that the File actions menu applies to the browse box with focus, which is the contract child browse box. Note also that I added the ‘count’ functionality, the result of which is displayed at the top of the window.

More interesting is the Columns menu. In this menu, the end user can define column presets (Both field and sort settings) for the browse box with focus. Right clicking the Standard item reveals a popup menu, as in Figure 3.

Figure 3. The Columns popup menu

Now, the user can create a new column set, change the name, or delete the column set (if there is more than one). For the new and change options, the user can edit the name:

Figure 4. Adding a new column

Then, from the properties item, a window is shown in which the end user can select the fields to display as columns, the order of these fields in the list box, or the fields on which the list has to be sorted (with multiple sort levels possible). These settings are stored per column set. In the end-user version, the field names are replaced with a short description of the field, using the data dictionary short description field.

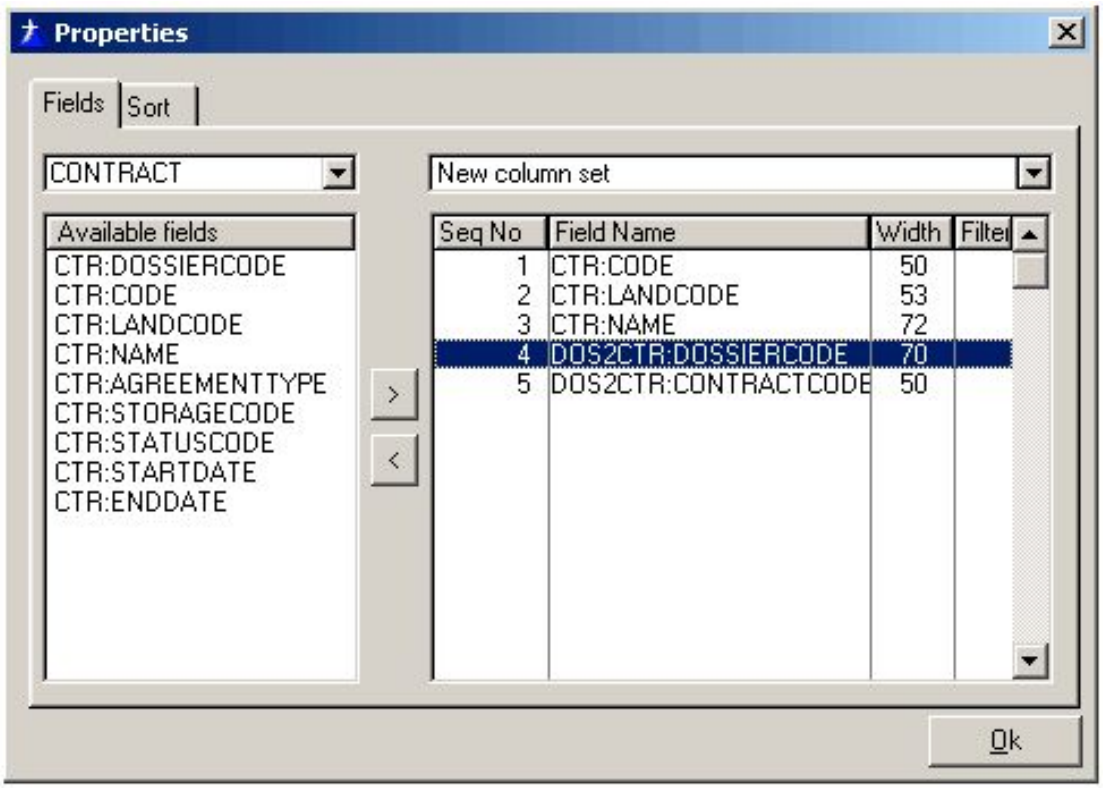


Figure 5. Setting the column fields

Removing all fields except CTR:Code and CTR:Name, and setting CTR:Code as the field to sort on, will result in the Contract browse in Figure 6.

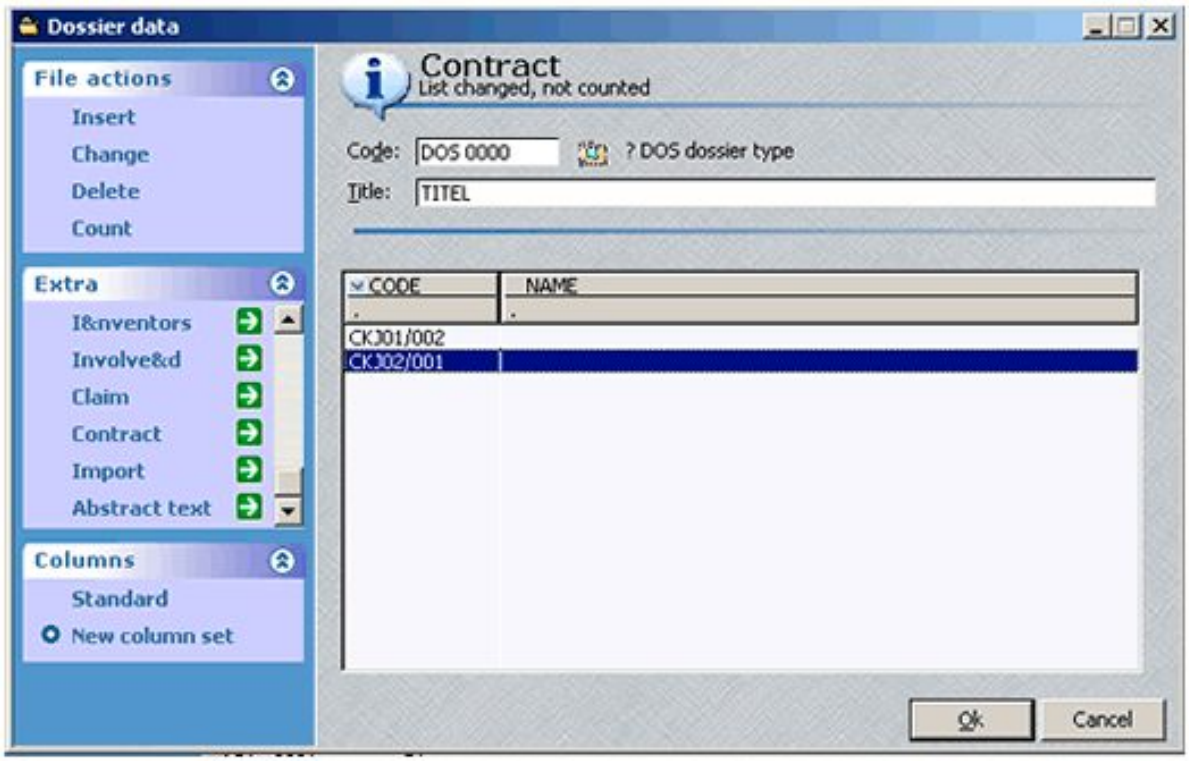


Figure 6. The Contract browse with the new column set.

Notice that the end-user now can switch very rapidly between the two column sets (Standard and New column set). An end user can use the column sets also for setting different sort orders while displaying the same information. An end-user can also use the column settings for saving filters, which I'll describe later.

The bullet icon more or less resembles a radio control, showing which set is selected, and indicating that only one item can be selected at a time.

Now, in every other window where there is a browse box showing contract records, these column settings can be used, and the items Standard and New column set will be shown to the user.

From the programmer's perspective, the browsebox is given a BrowseID, which should be unique for every unique view configuration. If the same parent and child files are used in a view, the same BrowseID can be used and the same column settings can be shown to the user.

In my implementation, column settings are unique for the window logon, so if someone else logs onto that machine, they can create their own settings.

Summary

Support can be reduced by building a consistent User Interface. An XP Menu can be used to build a consistent User Interface. To the programmer, a.User Interface using the XP Menu can easily be extended with other functionality. The XP menu can reduce the number of controls on a window, focusing the user on the actual data. A menu item can be focus driven, so that one menu item controls multiple other controls (list boxes). A menu can also be used for other purposes than mimicking a button. An example is the column set list provided in this article.

XP Menu Web Site: www.compad-software.com/uk/developer

Ronald van Raaphorst studied Chemical Engineering at the University of Enschede (UT), the Netherlands. But he found programming was more fun than designing a chemical plant, and when a roommate asked him to help start a software company, he found the choice easy to make. Ronald has used Clarion since 1994, beginning with Clarion 3 for DOS. Compad Software, which he co-owns, sells software to a small group of bakeries, he spends a considerable amount of time on the phone helping users, finding (and creating) new bugs, writing manuals, and of course programs. Ronald is in charge of developing Compad's products, and his colleague is on the road selling.

Reader Comments

Add a comment

Clarion Magazine

How To Ignore A Template

by Steven Parker

Published 2002-09-27

In a recent series of articles, I urged Clarion developers to "ignore" the templates and to call ABC methods directly. At the right, you will find links to those articles.

The foundation for this approach, of course, is the fact that the ABC templates are just wrappers for the ABC libraries and, therefore, if a template does not make something readily available (read "if the template doesn't hit you up side the head with an option"), just call the relevant ABC methods yourself, directly, in an embed. The good part of all this is that I can use OOP without having to *do* OOP.

I felt that I'd been fairly thorough documenting the methods I most often use directly, the methods I find address a good variety of business needs, what they do and how to use them. What I hadn't shown was how to discover which methods might be of interest.

The three most popular ways to discover which ABC method implements a particular functionality are:

1. Ask, usually on one of the SoftVelocity newsgroups. If you have the time to wait for a response, if a response is forthcoming at all (while not a frequent occurrence, some queries do go unanswered), this can be perfectly adequate. (If you do not have an immediate need to know, "lurking" in the newsgroups is an excellent way of learning your ABCs – it's how I picked up the basics.)
2. Read, comprehend and absorb the Application Handbook. I also call this the "osmosis approach" because the only way this is going to work for me is

The "Ignore" Articles

Reports: OOP, ABC and Ignoring Templates (Part 1)

Reports: OOP, ABC and Ignoring Templates (Part 2)

How To Ignore A

by osmosis.

3. Get the templates to tell you what methods they call, at what points they call them and how those methods are used. Yes, you can get the templates themselves to tell you what you want to know. Then you can ignore the templates.

Template: The Browse Reconsidered

Let me show how this last technique works by showing you how I discovered the methods I discussed in the previous articles.

Setting Keys with `AddSortOrder`

In Reports: OOP, ABC and Ignoring Templates (Part 2) I discussed the `AddSortOrder` method for dynamically selecting a key for a report or browse. How does one discover that it is the `AddSortOrder` method that the ABC libraries use to set keys?

This one is simple. Take a procedure, any procedure, and select a key for the primary file, as in Figure 1.

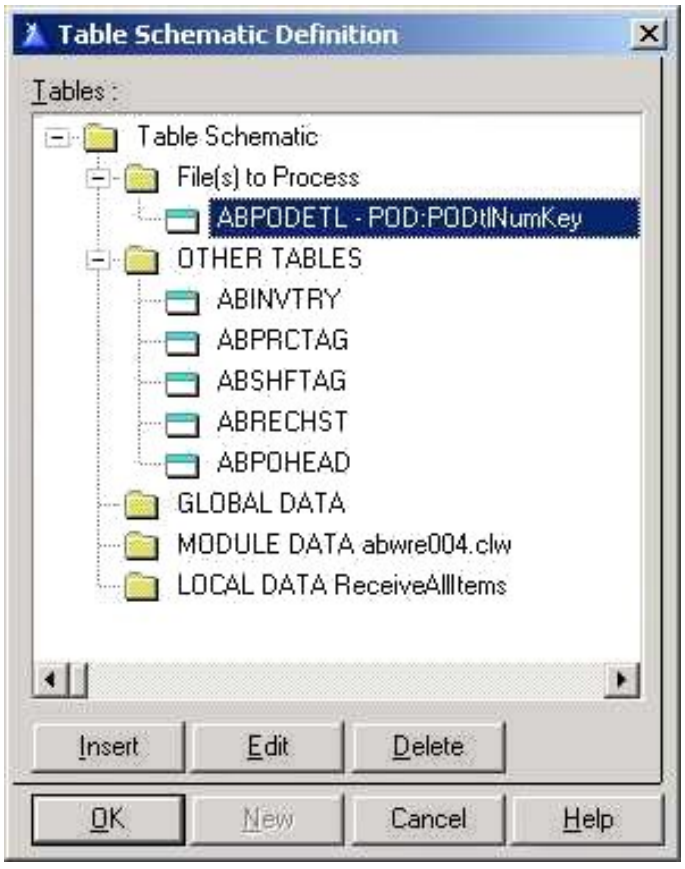


Figure 1. Standard key selection in File Schematic

Okay, not just "any procedure" will do, but a procedure like a browse, report or process where a key is meaningful. Save the procedure and return to the application tree. Right click the procedure and select "Source." This shows all of the source code that could be generated by the

template.

Now, locate the key you selected. You will see something like Figure 2.

```
! [Priority 8400]

ProgressMgr.Init(ScrollSort:AllowAlpha+ScrollSort:All
ThisProcess.Init(Process:View, Relate:ABPODETL, ?Prog
ThisProcess.CaseSensitiveValue = FALSE
ThisProcess.AddSortOrder(POD:PODtlNumKey)
ThisProcess.AddRange(POD:Number,POH:Number)
ThisProcess.SetFilter('POD:StoreID = POH:StoreID')
ProgressWindow{Prop:Text} = 'Processing Records'
?Progress:PctText{Prop:Text} = '0% Completed'
SELF.Init(ThisProcess)
?Progress:UserString{Prop:Text}=''
SELF.AddItem(?Progress:Cancel, RequestCancelled)
! Process field templates
! [Priority 8550]
this is the next embed
SEND(ABPODETL,'QUICKSCAN=on')
! [Priority 8800]

! Prepare Alert Keys
SELF.SetAlerts()
```

Figure 2. Generated code for key

Note the relationship between the template prompt and the generated code:

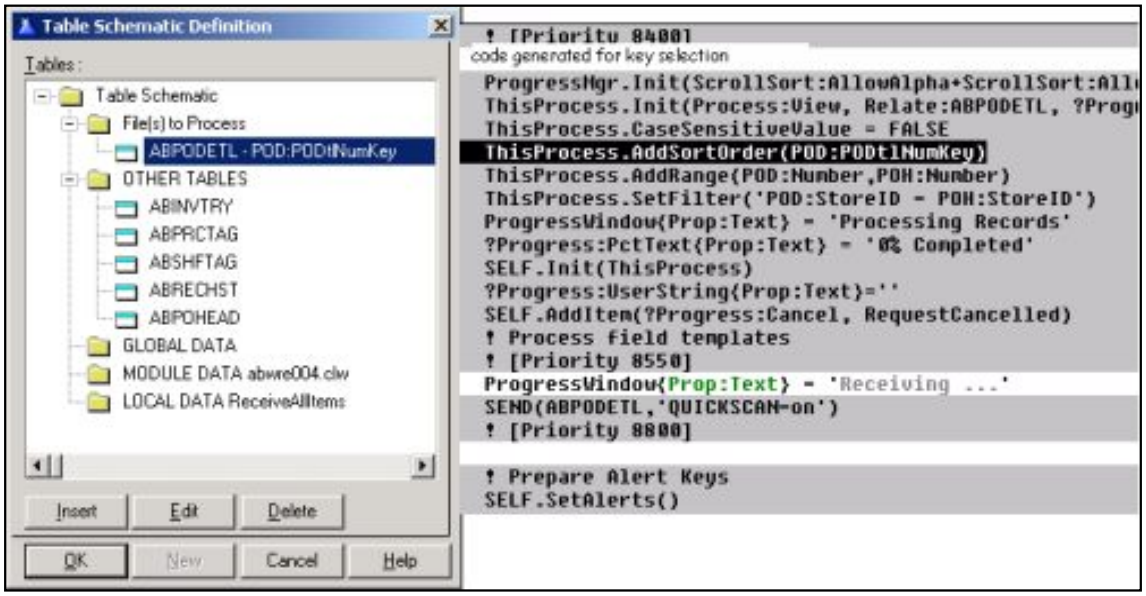


Figure 3. Template prompt and its corresponding code

That is how I discovered `AddSortOrder` and how I discovered the optimal embed for calling the method myself (the one immediately after the spot in which the template generates it).

At this point, remove the key from the File Schematic and add the code you actually want in INIT, 8550. (When using the Source view, I typically leave a bookmark behind in the embed I will be using, something like "`!here, dummy`.")

Setting filters with SetFilter

Figuring out how to do simple filters is almost exactly the same as discovering `AddSortOrder`. First, an entry in the template's Filter prompt:

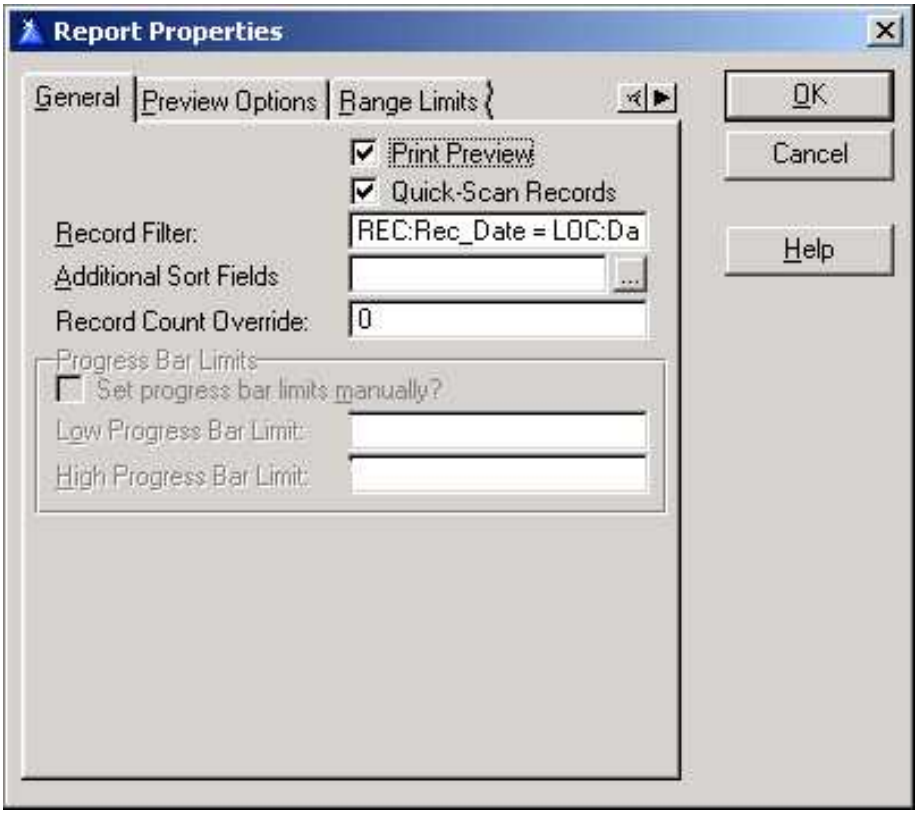


Figure 4. Filter prompt

Then check the generated code:

```
ThisReport.Init(Process:View, Relate:ABRECHST, ?Progres
ThisReport.AddSortOrder()
ThisReport.SetFilter('REC:Rec_Date = LOC:Date')
SELF.AddItem(?Progress:Cancel,RequestCancelled)
SELF.Init(ThisReport,Report,Previewer)
?Progress:UserString{Prop:Text}=''
Relate:ABRECHST.SetQuickScan(1,Propagate:OneMany)
! Process field templates
! [Priority 8505]

SELF.Zoom = PageWidth
Previewer.SetINIManager(INIMgr)
```

Figure 5. Code generated for a filter

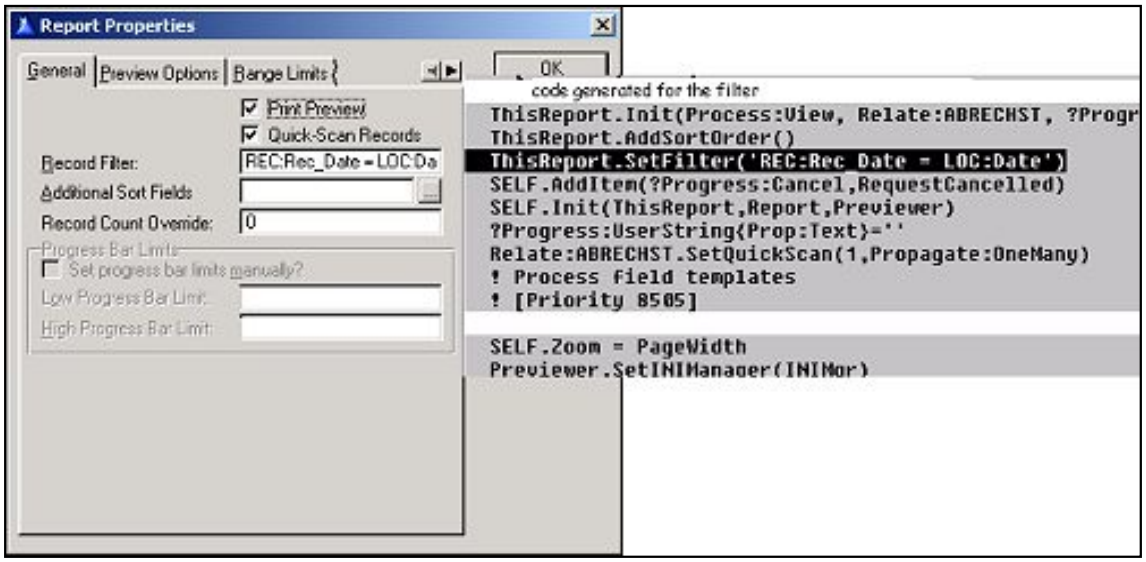


Figure 6. Relationship of template prompt to generated code

The *only* additional thing required to make extremely complex filters, using multiple `SetFilter` statements, is looking up `SetFilter` in the on-line help (just highlight the word "SetFilter" and press F1). Doing so shows that `SetFilter` is an overloaded procedure (it has two distinct prototypes). There is a variant that takes a second parameter and this second parameter allows concatenation of multiple `SetFilter` statements (and removal of empty filter expressions).

Sorting without keys: AppendOrder

In Reports: OOP, ABC and Ignoring Templates (Part 2) I also showed how `AppendOrder` allowed me to sort browses, reports and processes without using any key from the Dictionary (`AppendOrder`'s primary use is to add nodes to an existing key but it *can* be used on its own, as well).

`AppendOrder` turns out to be the ABC method called by the templates' Additional Sort Fields prompt (I used the template's Help button to understand what this prompt accomplishes).

Again, make an entry in the Additional Sort Fields prompt:

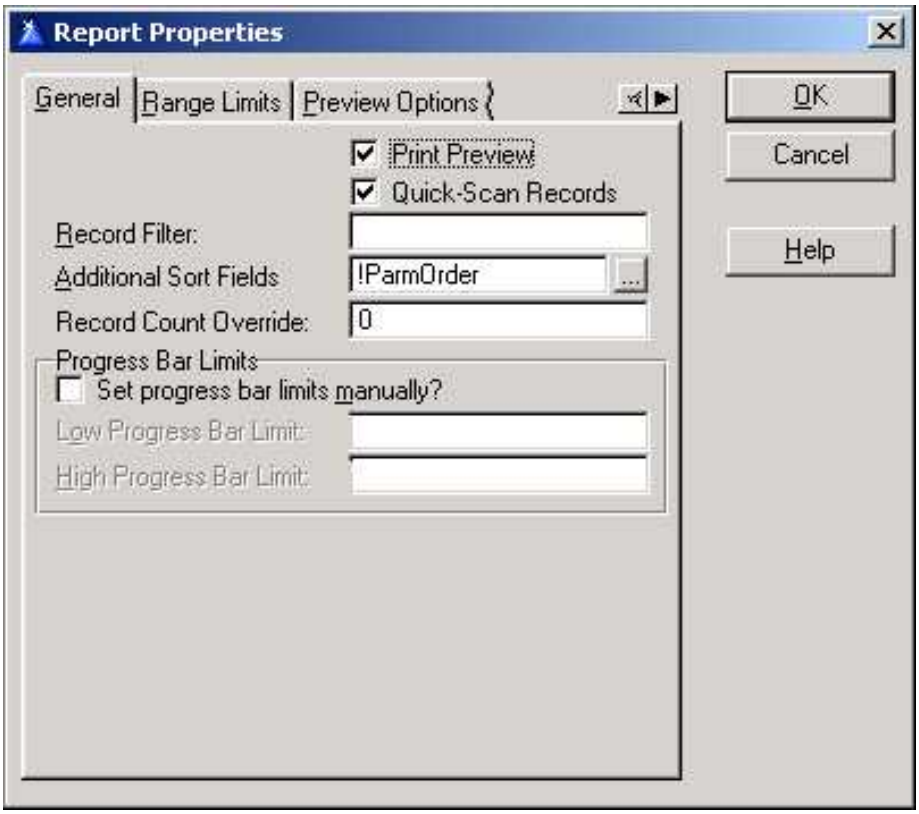


Figure 7. Additional Sort Fields prompt:

Find the generated code:

```
ParmOrder =Clip(Left(ParmOrder))
ProgressMgr.Init(ScrollSort:AllowNumeric,)
ThisReport.Init(Process:View, Relate:ABRODETL, ?Progres
ThisReport.AddSortOrder(ROD:RONumKey)
ThisReport.AddRange(ROD:RONumber,ROH:Number)
ThisReport.AppendOrder(ParmOrder)
SELF.AddItem(?Progress:Cancel,RequestCancelled)
SELF.Init(ThisReport,Report,Previewer)
?Progress:UserString{Prop:Text}=''
Relate:ABRODETL.SetQuickScan(1,Propagate:OneMany)
! Process field templates
! [Priority 8505]
```

Figure 8. Generated code for Sort Fields

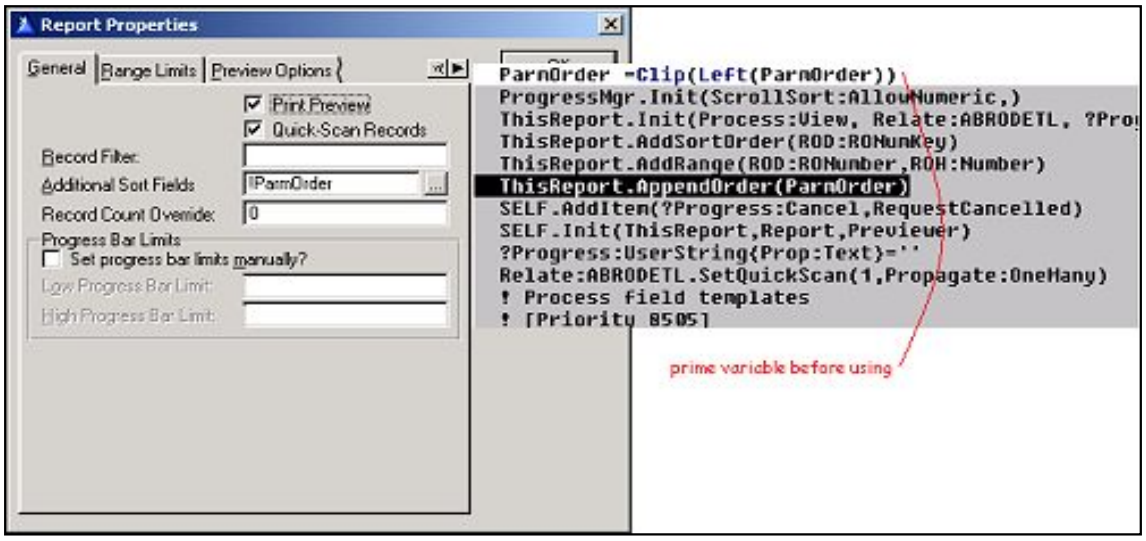


Figure 9. Relationship of template prompt to generated code

However, if you check the Template Help on "Additional Sort Fields," you may notice that it states:

> Specify fields to sort on in addition to any Key specified ... by typing in an ORDER expression list

There are two things that caught my attention here.

First, "in addition to any Key." This implies that a key is not required. Therefore, I could declare a primary file in a procedure without specifying a key and use `AppendOrder` to create whatever sort order I wanted. A quick test verified that this was the case.

Second, the Template Help refers to an expression, a comma delimited list of fields. It does *not* mention using variables. And you will note that I used a variable in Figure 7.

From looking at the generated code in Figure 8, I knew that `AppendOrder` was being called. So, I checked the on-line help for `AppendOrder`. Lo and behold, `AppendOrder` supports variables!

Et, voila!

(If you discover a template prompt that does not act correctly on a variable where the called method supports variables, you have a template bug. Report it.)

The only thing I must do is to ensure that the variable is primed before it is used (see the first line of code in Figure 8). The Filter prompt (Figure 4) can also take a variable, again, so long as it is primed before use and, in the case of filters only, `BIND`ed.

(Technically, you can do SetFilter any time before it is actually used. That is, SetFilter can be called any time before ApplyFilter is called. Similarly, AddSortOrder and AppendOrder can be called any time before SetSort is called. ApplyFilter and SetSort are the ABC methods which actually implement the requests made by AddSortOrder, AppendOrder and SetFilter. It's up to you if you want to research the latest possible point to call these methods; for the vast majority of needs, the "default" embed is perfect.)

SetOrder

The SetOrder method allows the same functionality as AppendOrder without a previous key declaration. In other words, you can create a completely *ad hoc* sort order for a browse, report or process. SetOrder has the advantage of being less restrictive than AppendOrder. Where AppendOrder must be called at certain points, SetOrder can be called much later. Thus, SetOrder *may* allow additional flexibility in procedure design.

AddSortOrder , SetFilter and AppendOrder occur late enough in the INIT method that collecting user information for the filter or order can be comfortably completed before these methods are called. At least for reports and processes, most browses, too.

But what if you want to be able to reset sorts and filters on the fly (in a browse, obviously)? You can use SetFilter this way but AddSortOrder and AppendOrder cannot, at least not easily.

You can use SetOrder just as you use SetFilter:

```
LOC:SortOrder = <new value>
BRW1.SetOrder(LOC:SortOrder)
ThisWindow.Reset(1)
Select(?Browse:1)
```

What SortOrder adds to the toolbox is flexibility.

Unfortunately, none of the knowledge acquisition methods mentioned above will provide information about this method. Looking at the example apps (specifically, PEOPLE.APP) does.

Summary

Using the template not only tells me many, if not most, of the methods I need to use but also shows me where I need to use them. Sometimes tracing the code generated by the templates isn't enough and I have to use the Template Help or general Help.

The Help tells me if a variable is acceptable and, in turn, that affects my design (I can't very well use a variable until it has a value, so data collection has to be finished before the embeds where the methods are called). Help also tells me about variations and limitations on the method.

After looking at the generated code and Help, I'm even (somewhat) prepared to examine and trace the code in the class libraries themselves. And, if the template plus Help approach doesn't help you figure out what you want, you can always read the manuals.

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

Reader Comments

Add a comment

The Clarion Challenge - ContainsMatch

Published 2002-09-27

Gordon Smith has proposed a Clarion Challenge, inspired by some code he came across recently. The problem? Write a string matching procedure which conforms to the following definition:

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

The procedure should return true when `txt` contains all the characters within `chars` *in any order*. The `noCase` parameter, when true, indicates that the comparison should be case insensitive.

For example:

- `ContainsMatch('Gordon Smith', 'hton')` returns `true`
- `ContainsMatch('Gordon Smith', 'htan')` returns `false`

For the purposes of this challenge, you only need to deal with 0-9, a-z and A-Z. Ignore spaces, i.e. do not consider a space in both strings a match. Also do not concern yourself with null characters, punctuation, or any other character not explicitly mentioned in the requirements. Yes, those are possible real-world issues, but the test here is how you would test for a match, not how you would anticipate all possible uses of the function.

So sharpen your typing fingers, slam out your slickest code, and send your entry to editor@clarionmag.com. Submissions will be featured in an upcoming issue of Clarion Magazine, and will be judged by Gordon.

Reader Comments

Add a comment

Gordon: What is you judging criteria? Least amount...
Gordon: What is you judging criteria? Least amount...
Efficency and readability (in that order). If the code...
I'm not clear on the specifications. It says ignore...
Having re-read the challenge (it is slightly different to...
Article says "do not concern yourself with...any other...
I would define ("Smith","h:t") as: B. True - must ignore...
In Gordon Smith's concept, he wrote '...all the characters...