

Clarion MAGAZINE

Clarion
Development
Resourcespublished by
CoveComm Inc.

clarion magazine

Good help isn't that hard to find.**\$6.²⁵/month**[Main Page](#)[Log In](#)
[Subscribe](#)
[Renewals](#)[Frequently Asked Questions](#)[Site Index](#)
[Article Index](#)
[Author Index](#)
[Links To Other Sites](#)[Downloads](#)
[Open Source Project](#)
[Issues in PDF Format](#)
[Free Software](#)[Advertising](#)[Contact Us](#)

Issue Index

January, 2000

[How To Handle Additional Sort Orders](#)

It's easy to add additional non-keyed sort orders to a browse. But how do you get around the problem of not being able to use locators on these sort orders? Jim Halpin shows the way.
(Jan 4,2000)

[The SQL Answer Cowboy](#)

The SQL Cowboy rides into town and answers your questions on SQL and Clarion.
(Jan 4,2000)

[Alphabetical Author Index](#)

Looking for articles by a particular author? Check out our alphabetical Author Index.
(Jan 4,2000)

[Alphabetical Article Index](#)

Know the name of an article but not where to find it? Look in our alphabetical article index.
(Jan 4,2000)

[Using MS Word With OLE: The Easiest Way](#)

Clarion OLE support for MS Word got you down? George Petrov builds on Jim Kane's OLE classes and creates a way to directly control Word from a Clarion application.
(Jan 11,2000)

[WebBuilder Skeleton Basics: Which? When?](#)

Steve Parker begins a weekly series explaining the inner workings of Clarion web application skeletons.
(Jan 11,2000)

[Understanding OOP Interfaces](#)

David Bayliss explains the inner workings of the new INTERFACE statement, and compares Clarion's implementation to Java's.
(Jan 18,2000)

[WebBuilder Skeleton Basics II: Logos and Fonts](#)

In Part 2 of his weekly WebBuilder series Steve Parker explains how to control fonts and logo placement.
(Jan 18,2000)

[xBASE Y2K Driver Patches](#)

Earlier versions of Clarion have problems reading xBase files when the file header has been updated incorrectly by another program for dates greater than 1999. These unsupported patches are intended to allow Clarion to work with such xBase files. Use at your own risk.



(Jan 18,2000)

[Advertise in Clarion Magazine!](#)

Clarion Magazine is a cost-effective way to reach other Clarion developers, whether you're selling third party products or development services, or looking to recruit talent.

(Jan 18,2000)

[The Novice's Corner: Clarion Code 3](#)

In this installment Dave Harms adds a window to last issue's file utility, and explains how user actions are processed.

(Jan 25,2000)

[Skeleton Basics III: Colors and Backgrounds](#)

When you think of customizing a web page, colors and background images are probably among the first things that come to mind. In his 75th Clarion article (not all for Clarion Magazine), Steve Parker explains how it's done.

(Jan 25,2000)

[January 2000 News](#)

Clarion news, notes, and happenings from around the globe.

(Jan 25,2000)

[The Cranky Programmer](#)

Having survived Y2K, Cranky gets wound up about old code. Specifically, old code no one has any idea how to maintain.

(Jan 25,2000)

Copyright © 1999-2000 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca.

published by
CoveComm Inc.

clarion magazine
Good help isn't that hard to find. **\$6.²⁵/month**

[Main Page](#)

[Log In](#)
[Subscribe](#)
[Renewals](#)

[Frequently Asked Questions](#)

[Site Index](#)
[Article Index](#)
[Author Index](#)
[Links To Other Sites](#)

[Downloads](#)
[Open Source Project](#)
[Issues in PDF Format](#)
[Free Software](#)

[Advertising](#)

[Contact Us](#)

Feature Article

January, 2000

A Beginner's Guide To Additional Sort Order Navigation

By Jim Halpin

Clarion's standard browse lets you display several sort orders for a table in a list box, with each sort order on a separate tab. Each tab can also have a Locator field, which lets the end user quickly locate records matching the Locator field entry. Clarion also lets you set up and display sort orders not based on an existing key. If you use this option, you lose the ability to provide a Locator field, since the Locator must be based on a key.

If you want to provide a non-key based sort order, how can you provide a way for your users to locate the records they need?

Choice #1 – Add A Key

One answer is to redesign the database, adding another key to the table and eliminating the need to specify an additional (non-key) sort order. Sometimes this is easily accomplished, but often it is impractical or undesirable.

A Little Background

In a true relational database design, one primary key is required for every table. A primary key is a unique (no duplicates), required (no nulls) index for the table. In this age of normalized data, SQL and client-server environments, a frequent recommendation is to design every table with a primary key that cannot be altered by the end user. By making the primary key a read-only, auto-populating field, you have complete control over the key values. You have the ability to "AutoNumber" the field, or substitute your own numbering scheme if this provides a better fit for the target environment. All of this can be hidden from the end user.

In this simple example there are two related tables, one for vehicles, and one for events that can happen to those vehicles:

Table	Vehicle	Events

- [How To Handle Additional Sort Orders](#)
(Jan 4,2000)
- [The SQL Answer Cowboy](#)
(Jan 4,2000)
- [Alphabetical Author Index](#)
(Jan 4,2000)
- [Alphabetical Article Index](#)
(Jan 4,2000)





Primary Key	VehID	EventID
Foreign Key		VehID
Other Fields	VehNumber	EventDescription
	VehicleDescription	EventDate

VehID and EventID (primary keys) are both populated behind the scenes with unique, never-changing values. The end user of the software doesn't even need to know that these fields exist.

As you establish the relationships between the tables in a database, each table's primary key can serve as a foreign key pointing to an entry in another table. In this example, each Events table entry contains a VehID that points to an entry in the Vehicle table. There is a one-to-many relationship between the Vehicle table and the Events table: each vehicle can have many events.

The end users can populate the Vehicle table by providing a VehNumber and Description of their own design. I don't need to specify any particular format for the VehNumber field, because I'm not going to use it as an index for the Vehicle table. To the end users, the VehNumber is the field they'll use to locate the record they need. In this design, VehNumber is just like any other non-key field.

When the end users populate data into the Events table, I'll force them to select a Vehicle entry from a list of VehNumbers. Internally, I'll store the VehID field that matches the VehNumber selected by the end user. The VehID field in the Events table is a foreign key, related to the Vehicle table primary key.

Now remember, VehID is the link between an Event and a Vehicle, but the end user thinks the link is VehNumber. They expect to be able to view the Events records sorted by VehNumber, and to use the VehNumber field to locate an Events record.

With choice #1, I can add a key to the Events table using the VehNumber field. If I make VehNumber the primary key of the Vehicle table, VehID is not necessary and can be eliminated in both tables. VehNumber would then be a foreign key field in the Events table, and a Clarion browse tab based on the VehNumber key will support a locator field.

The basic problem with this approach is that you give the end user access to (and the ability to change) the primary key of the Vehicle table. You want to keep the primary key hidden from the end user, and you want to populate it with a value of your choosing to prevent the user from altering data used to establish relationships between tables.

Choice #2 – Query

The standard Clarion browse offers the end user a Query option, based on fields displayed in the list box. This works great, but it is a lot more work than using a locator field. This choice is a good option, made better by adding it to Choice #3.

Choice #3 – A Pseudo-Locator Filter combination

In addition to the Query button, why not give the end user a familiar looking vehicle



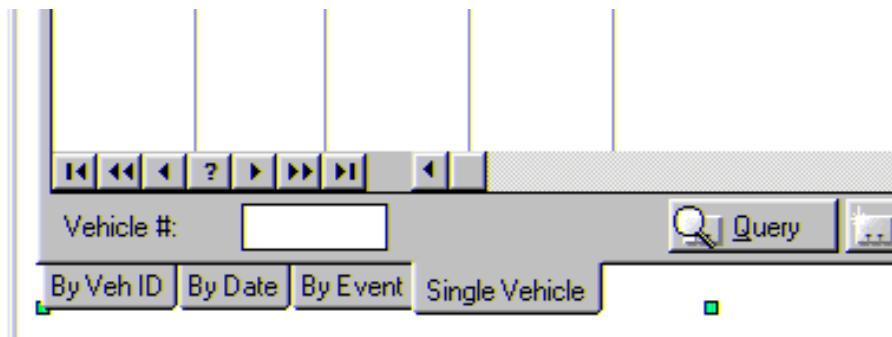
number entry field to locate and display records in the Events table?

Here are the steps needed:

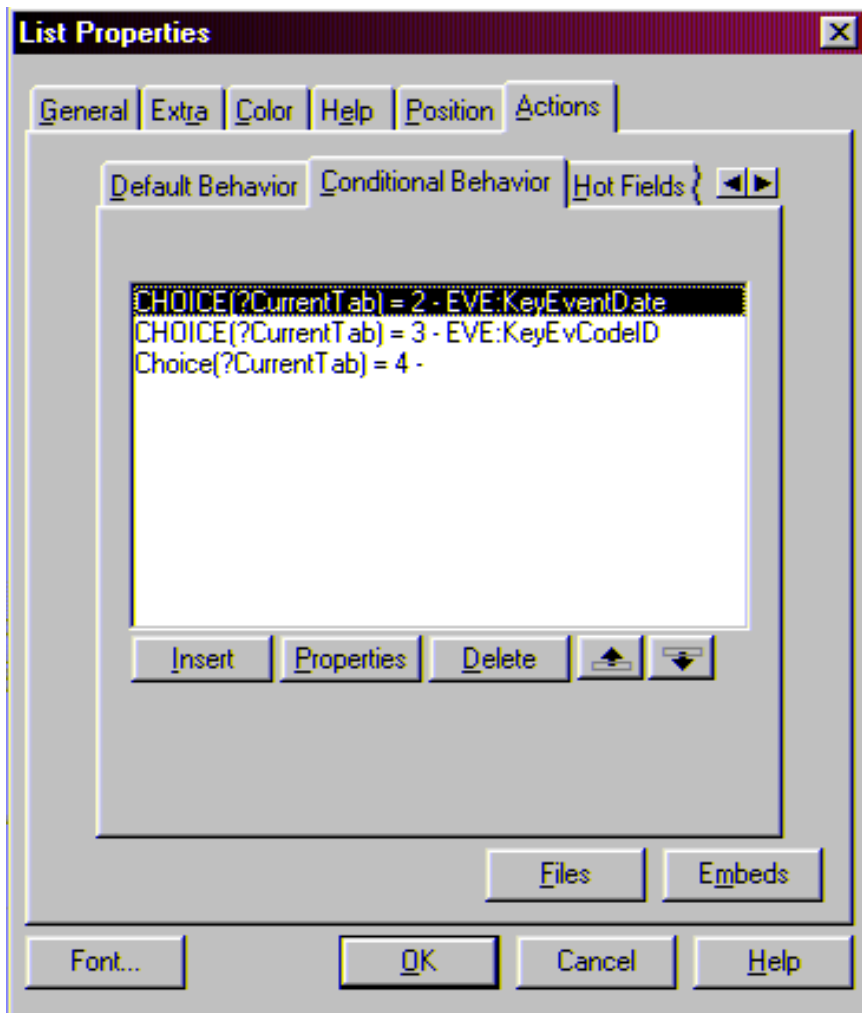
- Add a local variable to the procedure - `ThisVehicle`
- Add a tab to the browse list box
- Name the tab 'Single Vehicle'
- Add a string field to the new tab (Single Vehicle) and change the text to 'Vehicle #:'
- Populate the local variable (`ThisVehicle`) to the right of the string field
- On the Actions tab of the List Box properties, add the Conditional Behavior for the new tab
- Do not select a key for the sort order
- Add an additional sort order field – `VEH:VehNumber`
- Add a filter to limit records displayed based on the local variable (`ThisVehicle`)
- Use an embed to refresh the view when the Entry control (`ThisVehicle`) is accepted.
- Use an embed to skip the filter application until the end user enters a value.

The Details

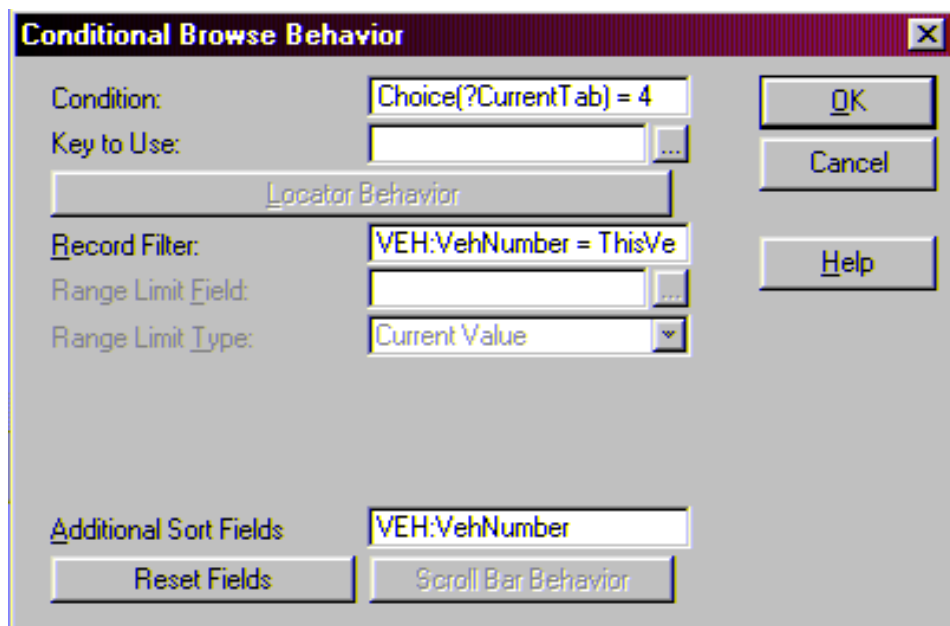
First, add a local variable to the browse procedure. Make it the same type of variable as the field used in the additional sort field – in this case, `VehNumber` is a string with a picture of `@s10`, so `ThisVehicle` will also be a string with a length of 10.



Next, add a tab to the browse list box, and name it 'Single Vehicle'. On this tab, place a string with 'Vehicle #:' as the text, and populate the local variable to the right of it. This looks like the typical Locator entry control.



On the Actions tab of the List Properties, go to the Conditional Behavior tab and insert a new condition where Choice(?CurrentTab) equals the new tab number (in the above example, Choice(?CurrentTab)=4).



Notice that the Key to Use entry is blank; there is no key available. Instead, specify VEH:VehNumber as an Additional Sort Field. Because there is no key, the Locator Behavior button is disabled. Remember that Locators can only be used with sort orders

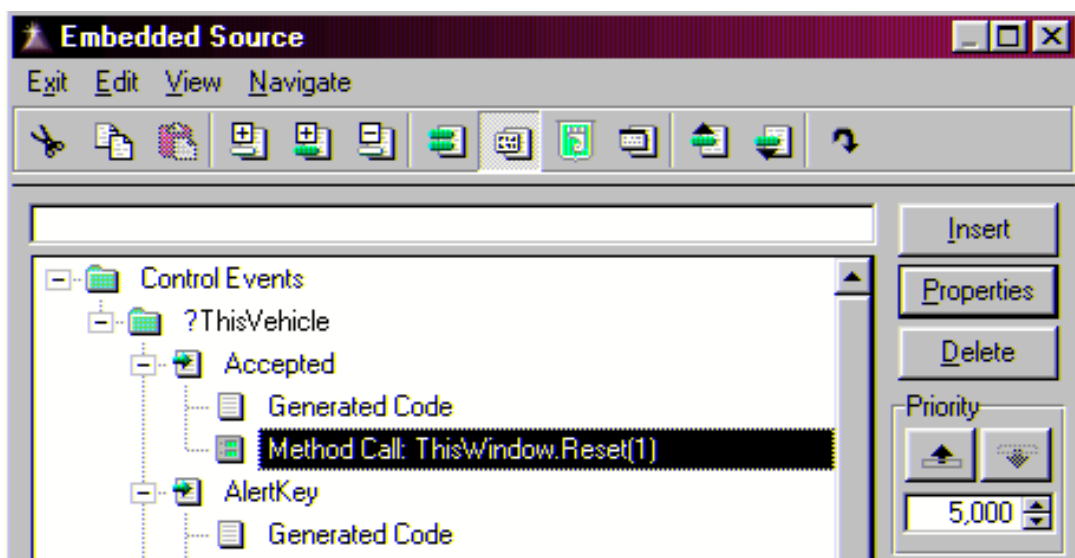
based on a key.

Next, set up a Record Filter. Type in `VEH:VehNumber = ThisVehicle`. The browse list box tab will only display records from the Events table where the related Vehicle table's record has a VehNumber that matches the contents of the local variable named `ThisVehicle`. Note: of course, you can apply different filter expressions to obtain different results.

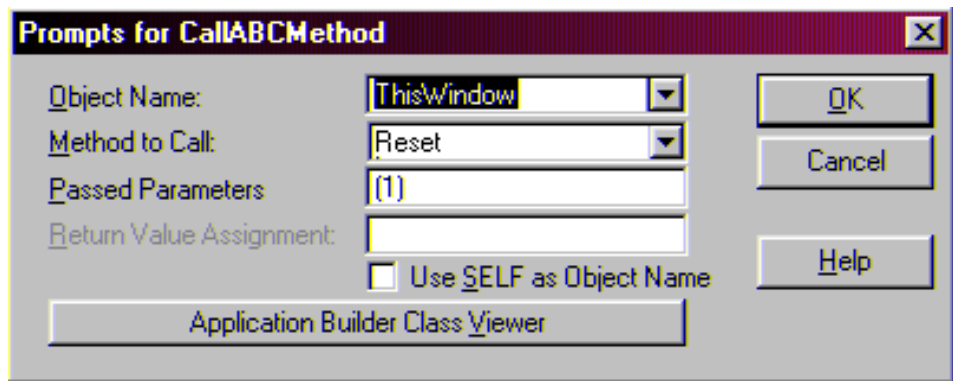
Remember to `BIND` any variables used in a filter expression.

When the end user first opens up the browse window, the Single Vehicle tab will contain all of the records in the Events table, sorted by VehNumber. The last step (adding an embed to skip the filter application unless the end user has supplied a value in `ThisVehicle`) makes it work this way. Without that embed, the filter would be applied and the list box would be empty. That's because the local variable is controlling the records displayed, and the end user hasn't given `ThisVehicle` a value yet. The end user types a value into the Entry control (the one that looks like a Locator field, and populates the local variable), and the browse list box can filter the records displayed to match that value.

The next step is to tell the procedure to refresh the contents of the Single Vehicle tab – that is, display the records requested by the filter. Without telling the procedure to refresh the display, the end user would need to click on another tab, then click back on the Single Vehicle tab in order to see the desired results. Use an embed to accomplish this.



There are several different ways to get the desired results, but a simple choice is to use the `ThisWindow.Reset` method call. At the embed point for the `ThisVehicle` Entry control, find the `Accepted` event and click the insert button. Select the `CallABCMethod` choice from the Class `ABC` – Application Builder Class Templates. You'll be presented with the following dialog:



Choose the `ThisWindow` object from the pull down list, choose the `Reset` method from the pull down list, and add `(1)` to the `Passed Parameters` field. This passed parameter `(1)` specifies an unconditional reset of the window.

Lastly, add the embed to skip the filter application until the end user enters a value in `ThisVehicle`. Find the `Browse on Events` local object (called `BRW6` or something similar), and move down to the `ApplyFilter` PROCEDURE, VIRTUAL embed point. Click on `Code`, then insert a source code embed that reads as follows:

```
If ThisVehicle = '' then Return.
```

This embed is entered into the code just before the `Parent Call` for the `ApplyFilter` procedure. If the variable is empty, the `Return` exits before the parent call is made, and the filter is not applied.

Summary

With these few simple steps, you've given the end users a choice. They can do a full query on the `Events` table, or go to the `Single Vehicle` tab. This tab lets them enter a value and see entries based on that value. They are applying a filter using a field that makes sense to them,

[Jim Halpin](#) is a self employed CPA and part time Clarion developer. He started programming in the late 70's, mostly with the Business BASIC language, making accounting related applications. He suspects he spends more time trying to keep up with Clarion changes and third party template upgrades than he does actually producing something useful.

Copyright © 1999-2000 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca.

Clarion MAGAZINE

Clarion
Development
Resources

published by
CoveComm Inc.

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

\$6.²⁵/month

[Main Page](#)

[Log In](#)
[Subscribe](#)
[Renewals](#)

[Frequently Asked Questions](#)

[Site Index](#)
[Article Index](#)
[Author Index](#)
[Links To Other Sites](#)

[Downloads](#)
[Open Source Project](#)
[Issues in PDF Format](#)
[Free Software](#)

[Advertising](#)

[Contact Us](#)

The SQL Cowboy

January, 2000

The SQL Answer Cowboy

Andy "Cowboy" Stapleton is the acknowledged Clarion SQL guru and a regular presenter at Clarion conferences around the world. His company, [Cowboy Computing Solutions](#), produces SQL templates and classes for Clarion.

If you have an SQL question for Andy, [click here](#).

Question:

How do you get PROP:SQL to work on a lookup and report? I want to do a lookup from a SQL database so that only a subset of records is returned. I understand it will dramatically reduce network traffic.

Yeoh Eng Loke

Answer:

If you are using the standard templates, the best method is using the Prop:SQLFILTER on the Lookup form. You can change the Prop:SQLFilter at a whim by using the browse with a passed parameter. To do this open your browse and insert (STRING) into the prototype field, and (pfilter) in the Parameters field. Now when calling the lookup you do it like this:

```
MyLookup('FieldA like <39>TEXAS%<39>')
```

In the Lookup procedure you will in an appropriate embed point have the following code.

```
MyView{Prop:sqlFilter}=pfilter
```

Remember that MyView is the Clarion-created view in the lookup and pfilter is the passed filter you sent. This will add your filter onto the Clarion-generated SQL statement and reduce your record set as desired.

Also be sure to upgrade to C5 if you haven't already. C4A and C4B both return the entire record on a file select, and while this may not be an entirely bad thing it does cause unnecessary traffic.

[How To Handle Additional Sort Orders](#)
(Jan 4,2000)

[The SQL Answer Cowboy](#)
(Jan 4,2000)

[Alphabetical Author Index](#)
(Jan 4,2000)

[Alphabetical Article Index](#)
(Jan 4,2000)



Cowboy



Question:

Is a VIEW structure generally an efficient approach for accessing a SQL back end? I have manually coded a view, including ORDER and FILTER options, and I am simply going one through the view, forwards, only once, reading the values and using them. The records themselves are not displayed, and the user cannot scroll backwards through the list or anything like that. (This isn't a browse - more something like a report). Sort of like a SQL Select - I want to get a reasonably generic method (which a view is) but which is also reasonably efficient.

Bruce Johnson

Answer:

The view structure is for more complicated items the best method to use, as it allows you to skip fields in the dictionary structure and also join additional tables together.

For example, think of Names and Address tables. The address table is a child table to names, and what you want to do is get the billing address on screen.

```

Create view NameAddr          view(NAMES)
Project (NAM:Namesysid)
Project (NAM:Fname)
Project (NAM:Lname)
Join(Addr:NameKey,NAM:Namesysid)
Project (Addr:Address1)
Project (Addr:Address2)
Project (Addr:CITY)
Project (Addr:STATE)
Project (Addr:Zip)
          END

open(NameAddr)
IF Errorcode();Stop(Error()).

NAMEADDR{Prop:SQL}=|
  'Select NAM.Namesysid,NAM.Fname,NAM.Lname, '&|
  'Addr.Address1,Addr.Address2,Addr,CITY,Addr, '&|
  'STATE,Addr:Zip'&|
  ' from '&NAME(NAMES)&' nam, '&NAME(ADDRESS)&'addr '&
  ' where Addr.Namesysid = NAm.Namesysid'
If Errorcode();Stop(FileError()).
Next(NameAddr)

```

Now you can be flexible; the only item you have to remember is the fields still have to be in FILE/FIELD order for the clarion record buffer to be accurate.

Cowboy

Question:

Andy, how do I count the number of records returned by a SELECT statement?

Perplexed in Peoria

Answer:

You can return the number of records easily by using a DUMMY file. Create a dummy file in your database:

```

Create table dummy (
  dummycounter    integer

```



```
);
```

Create the same in your dictionary. Now when you wish the correct number of records you return it into the dummy file. For example, I might want to return the number of addresses in Texas for Zipcode 78833.

```
Dummy{Prop:Sql}='Select Count(*) '&|  
  'from '&NAME(AddressTable) '&|  
  ' where State = <39>TX<39>' &|  
  ' and zipcode = '&78833  
If Errorcode();Stop(FileError()).  
Next(dummy)  
ScreenCounter = Dum:DummyCounter  
Display
```

The NAME(ADDRESSTABLE) is a clarion function that returns the external name of the dictionary, this is needed if you use multiple accounts on the database for login. The <39> is the ASCII equivalent to a single quote. You can use the dummy table for more than one thing; it can also return any other integer you need back to the clarion program.

Cowboy

If you have an SQL question for Andy, [click here](#).

Copyright © 1999-2000 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca.

Clarion MAGAZINE

Clarion
Development
Resources

published by
CoveComm Inc.

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

\$6.²⁵/month

[Main Page](#)

[Log In](#)
[Subscribe](#)
[Renewals](#)

[Frequently Asked Questions](#)

[Site Index](#)
[Article Index](#)
[Author Index](#)
[Links To Other Sites](#)

[Downloads](#)
[Open Source Project](#)
[Issues in PDF Format](#)
[Free Software](#)

[Advertising](#)

[Contact Us](#)

Clarion Magazine Information

February, 2000

Author Index

Barnes, Carl

[Clarion 5.5 Preview](#)
(Nov 2, 1999)

Bayliss, David

[David Bayliss On The ErrorClass](#)
(Feb 22, 1999)

[Snortaziw - Wizatrions from back to front](#)
(Mar 1, 1999)

[David Bayliss On The ConstantClass](#)
(Mar 19, 1999)

[David Bayliss on FieldClass](#)
(Apr 12, 1999)

[DAB Dissects The Clarion Challenge Results](#)
(Apr 19, 1999)

[DAB Dissects The Clarion Challenge Results](#)
(Apr 26, 1999)

[David Bayliss On The FileManager](#)
(May 17, 1999)

[David Bayliss On The FileManager Part 2](#)
(Jun 14, 1999)

[DAB - File Manager III](#)
(Jul 6, 1999)

[Propitious Memory Corruption](#)
(Aug 31, 1999)

[RelationManager Part 1](#)
(Sep 14, 1999)

[David Bayliss On The RelationManager - Part 2](#)
(Oct 26, 1999)

[ABC Design Series: The ViewManager Part 1](#)
(Nov 23, 1999)

[ABC Design Series: View Manager Part 2](#)
(Dec 7, 1999)

[Clarion Magazine Main Page](#)
(Feb 12, 1999)

[The Clarion Magazine FAQ](#)
(Feb 24, 1999)

[The Site Index](#)
(Feb 12, 1999)

[Subscribe to Clarion Magazine!](#)
(Feb 8, 1999)

[The Subscription Agreement](#)
(Feb 5, 1999)

[Refund Policy](#)
(Feb 8, 1999)

[Subscriber Services](#)
(Feb 8, 1999)

[Information For Authors](#)
(Feb 2, 1999)

[How To Contact Clarion Magazine](#)
(Feb 2, 1999)

[The Unofficial Clarion OOP Page](#)
(Feb 6, 1999)

[Lend Us Your Links](#)
(Feb 7, 1999)

[Getting Access To Clarion Magazine](#)
(Feb 7, 1999)

[Clarion Magazine's Automated Mailing Lists](#)
(Jan 31, 1999)

[Clarion Magazine Is Best Read With Verdana](#)
(Feb 7, 1999)

[Clarion Links By Category](#)
(May 17, 1999)

[Free Newsgroups For Clarion User Groups](#)
(May 10, 1999)

[Free Utilities: Class](#)



**How to
Put More
WOW!
into your
apps**

[Click Here](#)

[Understanding OOP Interfaces](#)

(Jan 18,2000)

Brewer, Simon

[ConVic '99 Clarion Conference](#)

(Oct 20,1999)

[Special Report: ConVic '99 Clarion Conference](#)

(Dec 7,1999)

Childers, Don

[Template Writing Made Easy](#)

(Dec 7,1999)

Cooke, James

[Edit-In-Place:Lights, Camera, Action...Take Two!](#)

(Feb 8,2000)

Creces, Gus

[The How And Why of WHO, WHAT, and WHERE](#)

(Mar 29,1999)

Eggen, Russell

[Demystifying The Debugger](#)

(May 25,1999)

[DevCon Latin America Pictures](#)

(May 25,1999)

Ferrett, Scott

[Freebie: How To Convert Your Database To SQL](#)

(May 3,1999)

Giles, Tom

[ABC Embeds Are Easy](#)

(Dec 21,1999)

Gilham, Bruce

[A Plan For Eliminating Bugs](#)

(Jul 6,1999)

[The Art Of Software Development: Analysis By Design](#)

(Nov 23,1999)

Guidroz II, Andrew

[Andrew's Kitchen - Sliding Into Cookery](#)

(Apr 26,1999)

[Guest Editorial: Andrew Guidroz II](#)

(Oct 14,1999)

[Andrew's Kitchen](#)

[Browser And Compile
Manager](#)

(Dec 7,1999)

[Alphabetical Author Index](#)

(Jan 4,2000)

[Alphabetical Article Index](#)

(Jan 4,2000)

[Clarion's Publication](#)

[Schedule](#)

(Feb 5,1999)

[Advertising Policy](#)

(Feb 7,1999)

[Open Source Code](#)

[Available Now](#)

(Mar 8,1999)

[The Masthead](#)

(Feb 15,1999)

(Oct 20, 1999)

Halpin, Jim

[How To Handle Additional Sort Orders](#)

(Jan 4, 2000)

Halsted, Pete

[Edit-In-Place CheckBoxes Done Right](#)

(Dec 21, 1999)

Harms, Dave

[ABC or Legacy: Which Templates Should You Use?](#)

(Feb 8, 1999)

[The Clarion Open Source Project](#)

(Feb 8, 1999)

[From The Publisher](#)

(Feb 8, 1999)

[The Clarion Advisor: Speed up your APP debugging with a PRJ](#)

(Feb 8, 1999)

[Interview: Roy Rafalco \(Part 1\)](#)

(Feb 15, 1999)

[Product Review: Xplore Templates](#)

(Feb 15, 1999)

[The Novice's Corner - Getting A Grip On Clarion](#)

(Feb 22, 1999)

[Interview: Roy Rafalco \(Part 2\)](#)

(Mar 1, 1999)

[The Clarion Advisor - Calculating Times](#)

(Mar 1, 1999)

[Open Source Public Information](#)

(Mar 8, 1999)

[The CCI Debug and Profiler Classes](#)

(Mar 8, 1999)

[The ABCs of OOP](#)

(Mar 19, 1999)

[The Novice's Corner - Understanding Templates And Embeds](#)

(Mar 19, 1999)

[Debugging Without The Debugger](#)

(Mar 29, 1999)

[Product Review: The Clarion Class Browser](#)

(Apr 12, 1999)

[Knowledge Bases On The Web - Feature Interview](#)

(Apr 19, 1999)

[The Novice's Corner - Designing Databases](#)

(May 3, 1999)

[Interview With Ragnar Hellspong](#)

(May 10, 1999)

[The ABCs of OOP - Part 2](#)

(May 17, 1999)

[Beta Tests: SFX Setup Builder](#)

(May 25, 1999)

[The Novice's Corner - Many-To-Many Relationships](#)

(Jun 14, 1999)

[The ABCs Of OOP - Part 3](#)

(Jun 28, 1999)

[Clarion Advisor: Debugging Tricks](#)

(Jun 28, 1999)

[Product Review: SearchFlash](#)

(Jul 20, 1999)

[Is It Six Months Already?](#)

(Aug 10, 1999)

[Interview: College Aid Calculator](#)

(Aug 17, 1999)

[Open Source Products Update](#)

(Aug 24, 1999)

[Industry Trends: How Important Is Java?](#)

(Sep 21, 1999)

[DevCon Weekend Edition](#)

(Sep 26, 1999)

[DevCon '99 Monday Overview](#)

(Sep 28, 1999)

[TopSpeed's New Direction: The Web](#)

(Sep 28, 1999)

[TopSpeed Headed For Java](#)

(Sep 29, 1999)

[DevCon Details: Web Edition 2 and iBuild@TopSpeed](#)

(Oct 6, 1999)

[DevCon Details: Welcome And Keynote Address](#)

(Oct 6, 1999)

[Seen And Heard At DevCon](#)

(Oct 6, 1999)

[DevCon Details: Richard Chapman On TopSpeed's Future Plans](#)

(Oct 14, 1999)

[Editorial: DevCon Wrapup](#)

(Oct 20, 1999)

[The Novice's Corner: Understanding Clarion Code](#)

(Oct 26, 1999)

[The Clarion Advisor:](#)

[Breaking Out Of Nested Loops](#)

(Nov 16, 1999)

[The Novice's Corner: Understanding Clarion Code](#)

(Dec 14, 1999)

[The Novice's Corner: Clarion Code 3](#)

(Jan 25, 2000)

Hebenstreit, Tom

[Review: MessageEx and SysAni](#)

(Feb 8, 2000)

[Product Review: Imaging Templates](#)

(Aug 17, 1999)

[The Cranky Programmer - Install THIS!](#)
(Sep 7, 1999)

[Product Review: IFT Server](#)
(Sep 21, 1999)

[Tool Talk: New Products At DevCon](#)
(Oct 14, 1999)

[Product Review: Clarion Source Search](#)
(Oct 26, 1999)

[Product Review: NiceTouch Dictionary/App Assistant](#)
(Nov 9, 1999)

[The Cranky Programmer: Nits And Bits](#)
(Dec 7, 1999)

[Product Review: ProDomus Translator Plus](#)
(Dec 14, 1999)

[The Cranky Programmer](#)
(Jan 25, 2000)

Johnson, Nik

[Working With Control Files II](#)
(Aug 10, 1999)

Kane, Jim

[The Other Way To Use OLE](#)
(Jul 20, 1999)

[The Other Way To Use OLE - Part 2](#)
(Sep 7, 1999)

[The Other Way To Use OLE - Part 3](#)
(Nov 2, 1999)

[A Class Wrapper For Files](#)
(Dec 21, 1999)

Marshall, Warren

[Australian DevCon Reports](#)
(Mar 19, 1999)

Morter, John

[Customizing Clarion5's Editor And Menus](#)
(Jul 13, 1999)

Mull, Stephen

[Freebie: Stephen Mull's Guide To Converting To MS-SQL](#)
(Nov 2, 1999)

O'Brien, Patrick

[The Clarion Advisor: Redirection Files](#)
(Sep 14, 1999)

Parker, Steven

[Skeleton Basics IV: List Variations](#)

(Feb 8, 2000)

[Don't Know, Do Care - A Philosopher Looks At OOP](#)

(Feb 15, 1999)

[NAME\(\) Comes Of Age](#)

(Mar 8, 1999)

[How ABC Handles Multiple Sort Orders](#)

(Apr 5, 1999)

[How ABC Handles Multiple Sort Orders \(Part II\)](#)

(May 10, 1999)

[How ABC Handles Multiple Sort Orders \(Part III\)](#)

(Jun 21, 1999)

[Working With Control Files I](#)

(Jul 27, 1999)

[Alias - Who Was That Masked File?](#)

(Aug 24, 1999)

[Clarion 5.5 Web Development Features](#)

(Nov 9, 1999)

[Relation Trees:](#)

[A Few Of The Finer Points](#)

(Nov 16, 1999)

[WebBuilder Skeleton Basics: Which? When?](#)

(Jan 11, 2000)

[WebBuilder Skeleton Basics II: Logos and Fonts](#)

(Jan 18, 2000)

[Skeleton Basics III: Colors and Backgrounds](#)

(Jan 25, 2000)

Petrov, George

[Using MS Word With OLE: The Easiest Way](#)

(Jan 11, 2000)

Pickus, Michael

[The Clarion Advisor: Changing Dictionaries](#)

(Aug 31, 1999)

[The Clarion Advisor - Listbox Styles](#)

(Dec 7, 1999)

Podger, David

[Detecting Crashes With DDE](#)

(Jun 28, 1999)

Ruby, Tom

[Presenting Many-To-Many Relationships](#)

(Nov 9, 1999)

[Presenting Many-To-Many Relationships:](#)

[Part 2](#)

(Nov 16, 1999)

Santos, Ross

[News Feature: Clarion Goes COM!](#)

(Apr 12, 1999)

Smith, Gordon

[Four DLLs And An Executable](#)

(Jun 7, 1999)

[The Clarion Advisor: Detecting Duplicate Records](#)

(Oct 26, 1999)

Sorzano, Troy

[How To Waste Time In The Newsgroups](#)

(Apr 19, 1999)

Stapleton, Andy

[Ask The SQL Answer Cowboy](#)

(Apr 5, 1999)

[The SQL Answer Cowboy](#)

(Apr 6, 1999)

[The SQL Answer Cowboy Answers!](#)

(Apr 26, 1999)

[The SQL Answer Cowboy](#)

(Jul 13, 1999)

[The SQL Answer Cowboy](#)

(Aug 17, 1999)

[The SQL Answer Cowboy](#)

(Dec 7, 1999)

[The SQL Answer Cowboy](#)

(Jan 4, 2000)

Teames, Larry

[Larry Teames on Reports](#)

(Apr 5, 1999)

[Larry Teames On Reports](#)

(May 10, 1999)

[Larry Teames On Reports](#)

(Jul 13, 1999)

[Larry Teames On Reports](#)

(Aug 31, 1999)

Thorley, John

[Australian DevCon Reports](#)

(Mar 19, 1999)

Tremblay, Pierre

[Sliders!](#)

(Jun 7, 1999)

Wells, Bruce

[The Clarion Advisor On Editor Colors](#)

(May 3, 1999)

Copyright © 1999-2000 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca.

Clarion MAGAZINE

Clarion
Development
Resources

published by
CoveComm Inc.

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

\$6.²⁵/month

[Main Page](#)

[Log In](#)
[Subscribe](#)
[Renewals](#)

[Frequently Asked Questions](#)

[Site Index](#)
[Article Index](#)
[Author Index](#)
[Links To Other Sites](#)

[Downloads](#)
[Open Source Project](#)
[Issues in PDF Format](#)
[Free Software](#)

[Advertising](#)

[Contact Us](#)

Clarion Magazine Information

February, 2000

Alphabetic Article Index

[A Class Wrapper For Files](#)

Clarion Magazine v1n11 - 12/01/1999
(Dec 21, 1999)

[A Plan For Eliminating Bugs](#)

Clarion Magazine v1n6 - 7/01/1999
(Jul 6, 1999)

[A Project Development Methodology](#)

Clarion Magazine v1n9 - 10/01/1999
(Oct 14, 1999)

[ABC Design Series: The ViewManager Part 1](#)

Clarion Magazine v1n10 - 11/01/1999
(Nov 23, 1999)

[ABC Design Series: View Manager Part 2](#)

Clarion Magazine v1n11 - 12/01/1999
(Dec 7, 1999)

[ABC Embeds Are Easy](#)

Clarion Magazine v1n11 - 12/01/1999
(Dec 21, 1999)

[ABC or Legacy: Which Templates Should You Use?](#)

Clarion Magazine v1n1 - 2/01/1999
(Feb 8, 1999)

[About Our New Look](#)

Clarion Magazine v1n11 - 12/01/1999
(Dec 7, 1999)

[Advertise in Clarion Magazine!](#)

Clarion Magazine v2n1 - 1/01/2000
(Jan 18, 2000)

[Advertising Packages](#)

Clarion Magazine v1n11 - 12/01/1999
(Dec 7, 1999)

[Alias - Who Was That Masked File?](#)

Clarion Magazine v1n7 - 8/01/1999
(Aug 24, 1999)

[Alphabetical Article Index](#)

Clarion Magazine v2n1 - 1/01/2000
(Jan 4, 2000)

[Alphabetical Author Index](#)

Clarion Magazine v2n1 - 1/01/2000
(Jan 4, 2000)

[Clarion Magazine Main Page](#)

(Feb 12, 1999)

[The Clarion Magazine FAQ](#)

(Feb 24, 1999)

[The Site Index](#)

(Feb 12, 1999)

[Subscribe to Clarion Magazine!](#)

(Feb 8, 1999)

[The Subscription Agreement](#)

(Feb 5, 1999)

[Refund Policy](#)

(Feb 8, 1999)

[Subscriber Services](#)

(Feb 8, 1999)

[Information For Authors](#)

(Feb 2, 1999)

[How To Contact Clarion Magazine](#)

(Feb 2, 1999)

[The Unofficial Clarion OOP Page](#)

(Feb 6, 1999)

[Lend Us Your Links](#)

(Feb 7, 1999)

[Getting Access To Clarion Magazine](#)

(Feb 7, 1999)

[Clarion Magazine's Automated Mailing Lists](#)

(Jan 31, 1999)

[Clarion Magazine Is Best Read With Verdana](#)

(Feb 7, 1999)

[Clarion Links By Category](#)

(May 17, 1999)

[Free Newsgroups For Clarion User Groups](#)

(May 10, 1999)

[Free Utilities: Class](#)

**How to
Put More
WOW!
into your
apps**

[Click Here](#)

[Andrew's Kitchen](#)

Clarion Magazine v1n9 - 10/01/1999
(Oct 20, 1999)

[Andrew's Kitchen - Sliding Into Cookery](#)

Clarion Magazine v1n3 - 4/01/1999
(Apr 26, 1999)

[Anniversary Subscription Special!](#)

Clarion Magazine v2n2 - 2/01/2000
(Feb 8, 2000)

[April News](#)

Clarion Magazine v1n3 - 4/01/1999
(Apr 26, 1999)

[Ask The SQL Answer Cowboy](#)

Clarion Magazine v1n3 - 4/01/1999
(Apr 5, 1999)

[Attend ETC. Win Stuff!](#)

Clarion Magazine v2n2 - 2/01/2000
(Feb 8, 2000)

[Australian DevCon Reports](#)

Clarion Magazine v1n2 - 3/01/1999
(Mar 19, 1999)

[Beta Tests: SFX Setup Builder](#)

Clarion Magazine v1n4 - 5/01/1999
(May 25, 1999)

[Call For Photos](#)

Clarion Magazine v1n5 - 6/01/1999
(Jun 7, 1999)

[Clarion 5.5 Preview](#)

Clarion Magazine v1n10 - 11/01/1999
(Nov 2, 1999)

[Clarion 5.5 Web Development Features](#)

Clarion Magazine v1n10 - 11/01/1999
(Nov 9, 1999)

[Clarion Advisor - Drive Free Space](#)

Clarion Magazine v1n6 - 7/01/1999
(Jul 6, 1999)

[Clarion Advisor: Debugging Tricks](#)

Clarion Magazine v1n5 - 6/01/1999
(Jun 28, 1999)

[Clarion Challenge String Parser Final Results](#)

Clarion Magazine v1n6 - 7/01/1999
(Jul 27, 1999)

[Clarion Links By Category](#)

Clarion Magazine v1n4 - 5/01/1999
(May 17, 1999)

[Clarion Magazine Best Read With Verdana](#)

Clarion Magazine v1n5 - 6/01/1999
(Jun 28, 1999)

[Clarion Magazine Holiday Schedule](#)

Clarion Magazine v1n11 - 12/01/1999
(Dec 21, 1999)

[Clarion Magazine Now Available In PDF Format](#)

Clarion Magazine v1n3 - 4/01/1999
(Apr 19, 1999)

[Clarion Magazine Version 2!](#)

Clarion Magazine v1n10 - 11/01/1999
(Nov 23, 1999)

[Clarion News - August 1999](#)

[Browser And Compile
Manager](#)

(Dec 7, 1999)

[Alphabetical Author Index](#)

(Jan 4, 2000)

[Alphabetical Article Index](#)

(Jan 4, 2000)

[Clarion's Publication](#)

[Schedule](#)

(Feb 5, 1999)

[Advertising Policy](#)

(Feb 7, 1999)

[Open Source Code](#)

[Available Now](#)

(Mar 8, 1999)

[The Masthead](#)

(Feb 15, 1999)

[Clarion Magazine v1n7 - 8/01/1999](#)
(Aug 31, 1999)

[Clarion News - May 1999](#)

Clarion Magazine v1n4 - 5/01/1999
(May 25, 1999)

[Clarion News, July 1999](#)

Clarion Magazine v1n6 - 7/01/1999
(Jul 27, 1999)

[ClarionMag Newsgroup Reminder](#)

Clarion Magazine v1n11 - 12/01/1999
(Dec 7, 1999)

[ConVic '99 Clarion Conference](#)

Clarion Magazine v1n9 - 10/01/1999
(Oct 20, 1999)

[Correction: Class And Wrapper For Handling Control Files](#)

Clarion Magazine v1n8 - 9/01/1999
(Sep 7, 1999)

[Customizing Clarion5's Editor And Menus](#)

Clarion Magazine v1n6 - 7/01/1999
(Jul 13, 1999)

[DAB - File Manager III](#)

Clarion Magazine v1n6 - 7/01/1999
(Jul 6, 1999)

[DAB Dissects The Clarion Challenge Results](#)

Clarion Magazine v1n3 - 4/01/1999
(Apr 19, 1999)

[DAB Dissects The Clarion Challenge Results](#)

Clarion Magazine v1n3 - 4/01/1999
(Apr 26, 1999)

[David Bayliss on FieldClass](#)

Clarion Magazine v1n3 - 4/01/1999
(Apr 12, 1999)

[David Bayliss On The ConstantClass](#)

Clarion Magazine v1n2 - 3/01/1999
(Mar 19, 1999)

[David Bayliss On The ErrorClass](#)

Clarion Magazine v1n1 - 2/01/1999
(Feb 22, 1999)

[David Bayliss On The FileManager](#)

Clarion Magazine v1n4 - 5/01/1999
(May 17, 1999)

[David Bayliss On The FileManager Part 2](#)

Clarion Magazine v1n5 - 6/01/1999
(Jun 14, 1999)

[David Bayliss On The RelationManager - Part 2](#)

Clarion Magazine v1n9 - 10/01/1999
(Oct 26, 1999)

[Debugging Without The Debugger](#)

Clarion Magazine v1n2 - 3/01/1999
(Mar 29, 1999)

[December 1999 News](#)

Clarion Magazine v1n11 - 12/01/1999
(Dec 21, 1999)

[Demystifying The Debugger](#)

Clarion Magazine v1n4 - 5/01/1999
(May 25, 1999)

[Detecting Crashes With DDE](#)

Clarion Magazine v1n5 - 6/01/1999

(Jun 28, 1999)

[DevCon '99 Monday Overview](#)

Clarion Magazine v1n8 - 9/01/1999
(Sep 28, 1999)

[DevCon '99 Special Subscription Offer Extended To October 15th!](#)

Clarion Magazine v1n9 - 10/01/1999
(Oct 14, 1999)

[DevCon Details: Richard Chapman On TopSpeed's Future Plans](#)

Clarion Magazine v1n9 - 10/01/1999
(Oct 14, 1999)

[DevCon Details: Web Edition 2 and iBuild@TopSpeed](#)

Clarion Magazine v1n9 - 10/01/1999
(Oct 6, 1999)

[DevCon Details: Welcome And Keynote Address](#)

Clarion Magazine v1n9 - 10/01/1999
(Oct 6, 1999)

[DevCon Latin America Pictures](#)

Clarion Magazine v1n4 - 5/01/1999
(May 25, 1999)

[DevCon Weekend Edition](#)

Clarion Magazine v1n8 - 9/01/1999
(Sep 26, 1999)

[Developers' Open Source Public License Version 1.0](#)

Clarion Magazine v1n5 - 6/01/1999
(Jun 21, 1999)

[Don't Know, Do Care - A Philosopher Looks At OOP](#)

Clarion Magazine v1n1 - 2/01/1999
(Feb 15, 1999)

[Edit-In-Place CheckBoxes Done Right](#)

Clarion Magazine v1n11 - 12/01/1999
(Dec 21, 1999)

[Edit-In-Place: Lights, Camera, Action...Take Two!](#)

Clarion Magazine v2n2 - 2/01/2000
(Feb 8, 2000)

[Editorial: DevCon Wrapup](#)

Clarion Magazine v1n9 - 10/01/1999
(Oct 20, 1999)

[February 2000 News](#)

Clarion Magazine v2n2 - 2/01/2000
(Feb 8, 2000)

[February News](#)

Clarion Magazine v1n1 - 2/01/1999
(Feb 22, 1999)

[Four DLLs And An Executable](#)

Clarion Magazine v1n5 - 6/01/1999
(Jun 7, 1999)

[Free Newsgroups For Clarion User Groups](#)

Clarion Magazine v1n4 - 5/01/1999
(May 10, 1999)

[Free Utilities: Class Browser And Compile Manager](#)

Clarion Magazine v1n11 - 12/01/1999
(Dec 7, 1999)

[Freebie: How To Convert Your Database To SQL](#)

Clarion Magazine v1n4 - 5/01/1999
(May 3, 1999)

[Freebie: Stephen Mull's Guide To Converting To MS-SQL](#)

Clarion Magazine v1n10 - 11/01/1999
(Nov 2, 1999)

[From The Publisher](#)

Clarion Magazine v1n1 - 2/01/1999
(Feb 8, 1999)

[Get On Our Mailing List](#)

Clarion Magazine v1n5 - 6/01/1999
(Jun 14, 1999)

[Guest Editorial: Andrew Guidroz II](#)

Clarion Magazine v1n9 - 10/01/1999
(Oct 14, 1999)

[How ABC Handles Multiple Sort Orders](#)

Clarion Magazine v1n3 - 4/01/1999
(Apr 5, 1999)

[How ABC Handles Multiple Sort Orders \(Part II\)](#)

Clarion Magazine v1n4 - 5/01/1999
(May 10, 1999)

[How ABC Handles Multiple Sort Orders \(Part III\)](#)

Clarion Magazine v1n5 - 6/01/1999
(Jun 21, 1999)

[How To Handle Additional Sort Orders](#)

Clarion Magazine v2n1 - 1/01/2000
(Jan 4, 2000)

[How To Waste Time In The Newsgroups](#)

Clarion Magazine v1n3 - 4/01/1999
(Apr 19, 1999)

[Industry Trends: How Important Is Java?](#)

Clarion Magazine v1n8 - 9/01/1999
(Sep 21, 1999)

[Interview With Ragnar Hellsping](#)

Clarion Magazine v1n4 - 5/01/1999
(May 10, 1999)

[Interview: College Aid Calculator](#)

Clarion Magazine v1n7 - 8/01/1999
(Aug 17, 1999)

[Interview: Roy Rafalco \(Part 1\)](#)

Clarion Magazine v1n1 - 2/01/1999
(Feb 15, 1999)

[Interview: Roy Rafalco \(Part 2\)](#)

Clarion Magazine v1n2 - 3/01/1999
(Mar 1, 1999)

[Is It Six Months Already?](#)

Clarion Magazine v1n7 - 8/01/1999
(Aug 10, 1999)

[January 2000 News](#)

Clarion Magazine v2n1 - 1/01/2000
(Jan 25, 2000)

[June 1999 News](#)

Clarion Magazine v1n5 - 6/01/1999
(Jun 28, 1999)

[Knowledge Bases On The Web - Feature Interview](#)

Clarion Magazine v1n3 - 4/01/1999
(Apr 19, 1999)

[Larry Teames on Reports](#)

Clarion Magazine v1n3 - 4/01/1999
(Apr 5, 1999)

[Larry Teames On Reports](#)

Clarion Magazine v1n6 - 7/01/1999
(Jul 13, 1999)

[Larry Teames On Reports](#)

Clarion Magazine v1n4 - 5/01/1999
(May 10, 1999)

[Larry Teames On Reports](#)

Clarion Magazine v1n7 - 8/01/1999
(Aug 31, 1999)

[March News](#)

Clarion Magazine v1n2 - 3/01/1999
(Mar 29, 1999)

[NAME\(\) Comes Of Age](#)

Clarion Magazine v1n2 - 3/01/1999
(Mar 8, 1999)

[New ClarionMag Link Buttons](#)

Clarion Magazine v1n11 - 12/01/1999
(Dec 7, 1999)

[New Improved Clarion Challenge!](#)

Clarion Magazine v1n5 - 6/01/1999
(Jun 21, 1999)

[New Open Source Products](#)

Clarion Magazine v1n7 - 8/01/1999
(Aug 10, 1999)

[News Feature: Clarion Goes COM!](#)

Clarion Magazine v1n3 - 4/01/1999
(Apr 12, 1999)

[November 1999 News](#)

Clarion Magazine v1n10 - 11/01/1999
(Nov 23, 1999)

[October 1999 News](#)

Clarion Magazine v1n9 - 10/01/1999
(Oct 26, 1999)

[Open Source Code Available Now](#)

Clarion Magazine v1n2 - 3/01/1999
(Mar 8, 1999)

[Open Source Products Update](#)

Clarion Magazine v1n7 - 8/01/1999
(Aug 24, 1999)

[Open Source Public Information](#)

Clarion Magazine v1n2 - 3/01/1999
(Mar 8, 1999)

[Open Source Update](#)

Clarion Magazine v1n9 - 10/01/1999
(Oct 20, 1999)

[Open Source Update](#)

Clarion Magazine v1n8 - 9/01/1999
(Sep 7, 1999)

[Open Source Update](#)

Clarion Magazine v1n8 - 9/01/1999
(Sep 14, 1999)

[Open Source Update: Date Fix & DDE Classes](#)

Clarion Magazine v1n11 - 12/01/1999
(Dec 14, 1999)

[Open Source Update: SQL Class](#)

Clarion Magazine v1n9 - 10/01/1999
(Oct 14, 1999)

[Photo Gallery](#)

Clarion Magazine v1n6 - 7/01/1999
(Jul 13, 1999)

[Presenting Many-To-Many Relationships](#)

Clarion Magazine v1n10 - 11/01/1999
(Nov 9,1999)

[Presenting Many-To-Many Relationships:](#)

[Part 2](#)

Clarion Magazine v1n10 - 11/01/1999
(Nov 16,1999)

[Problems With PDFs?](#)

Clarion Magazine v1n5 - 6/01/1999
(Jun 7,1999)

[Product Review: Clarion Source Search](#)

Clarion Magazine v1n9 - 10/01/1999
(Oct 26,1999)

[Product Review: IFT Server](#)

Clarion Magazine v1n8 - 9/01/1999
(Sep 21,1999)

[Product Review: Imaging Templates](#)

Clarion Magazine v1n7 - 8/01/1999
(Aug 17,1999)

[Product Review: NiceTouch Dictionary/App Assistant](#)

Clarion Magazine v1n10 - 11/01/1999
(Nov 9,1999)

[Product Review: ProDomus Translator Plus](#)

Clarion Magazine v1n11 - 12/01/1999
(Dec 14,1999)

[Product Review: SearchFlash](#)

Clarion Magazine v1n6 - 7/01/1999
(Jul 20,1999)

[Product Review: The Clarion Class Browser](#)

Clarion Magazine v1n3 - 4/01/1999
(Apr 12,1999)

[Product Review: Xplore Templates](#)

Clarion Magazine v1n1 - 2/01/1999
(Feb 15,1999)

[Propitious Memory Corruption](#)

Clarion Magazine v1n7 - 8/01/1999
(Aug 31,1999)

[Proposed Open Source License](#)

Clarion Magazine v1n2 - 3/01/1999
(Mar 19,1999)

[Publishing Schedule](#)

Clarion Magazine v1n6 - 7/01/1999
(Jul 13,1999)

[Relation Trees:](#)

[A Few Of The Finer Points](#)

Clarion Magazine v1n10 - 11/01/1999
(Nov 16,1999)

[RelationManager Part 1](#)

Clarion Magazine v1n8 - 9/01/1999
(Sep 14,1999)

[Renew Your ClarionMag Subscription And Save!](#)

Clarion Magazine v2n2 - 2/01/2000
(Feb 8,2000)

[Review: MessageEx and SysAni](#)

Clarion Magazine v2n2 - 2/01/2000
(Feb 8,2000)

[Search Clarion Magazine](#)

Clarion Magazine v1n5 - 6/01/1999
(Jun 14,1999)

[Search Clarion Magazine!](#)

Clarion Magazine v1n3 - 4/01/1999
(Apr 19,1999)

[Seen And Heard At DevCon](#)

Clarion Magazine v1n9 - 10/01/1999
(Oct 6,1999)

[September News](#)

Clarion Magazine v1n8 - 9/01/1999
(Sep 21,1999)

[Skeleton Basics III: Colors and Backgrounds](#)

Clarion Magazine v2n1 - 1/01/2000
(Jan 25,2000)

[Skeleton Basics IV: List Variations](#)

Clarion Magazine v2n2 - 2/01/2000
(Feb 8,2000)

[Sliders!](#)

Clarion Magazine v1n5 - 6/01/1999
(Jun 7,1999)

[Snortaziw - Wizatrons from back to front](#)

Clarion Magazine v1n2 - 3/01/1999
(Mar 1,1999)

[Special Report: ConVic '99 Clarion Conference](#)

Clarion Magazine v1n11 - 12/01/1999
(Dec 7,1999)

[Technology Poll Results](#)

Clarion Magazine v1n8 - 9/01/1999
(Sep 21,1999)

[Template Writing Made Easy](#)

Clarion Magazine v1n11 - 12/01/1999
(Dec 7,1999)

[The ABCs of OOP](#)

Clarion Magazine v1n2 - 3/01/1999
(Mar 19,1999)

[The ABCs of OOP - Part 2](#)

Clarion Magazine v1n4 - 5/01/1999
(May 17,1999)

[The ABCs Of OOP - Part 3](#)

Clarion Magazine v1n5 - 6/01/1999
(Jun 28,1999)

[The Art Of Software Development: Analysis By Design](#)

Clarion Magazine v1n10 - 11/01/1999
(Nov 23,1999)

[The CCI Debug and Profiler Classes](#)

Clarion Magazine v1n2 - 3/01/1999
(Mar 8,1999)

[The Clarion Advisor - Calculating Times](#)

Clarion Magazine v1n2 - 3/01/1999
(Mar 1,1999)

[The Clarion Advisor - Fast ASCII](#)

Clarion Magazine v1n4 - 5/01/1999
(May 25,1999)

[The Clarion Advisor - Listbox Styles](#)

Clarion Magazine v1n11 - 12/01/1999
(Dec 7,1999)

[The Clarion Advisor On Editor Colors](#)

Clarion Magazine v1n4 - 5/01/1999
(May 3,1999)

[The Clarion Advisor: Changing Dictionaries](#)

[Clarion Magazine v1n7 - 8/01/1999
\(Aug 31, 1999\)](#)

[The Clarion Advisor: Debug Redux](#)

Clarion Magazine v1n6 - 7/01/1999
(Jul 20, 1999)

[The Clarion Advisor: Detecting Duplicate Records](#)

Clarion Magazine v1n9 - 10/01/1999
(Oct 26, 1999)

[The Clarion Advisor: Redirection Files](#)

Clarion Magazine v1n8 - 9/01/1999
(Sep 14, 1999)

[The Clarion Advisor: Speed up your APP debugging with a PRJ](#)

Clarion Magazine v1n1 - 2/01/1999
(Feb 8, 1999)

[The Clarion Advisor: Topspeed Driver Error Codes](#)

Clarion Magazine v1n1 - 2/01/1999
(Feb 22, 1999)

[The Clarion Advisor:](#)

[Breaking Out Of Nested Loops](#)

Clarion Magazine v1n10 - 11/01/1999
(Nov 16, 1999)

[The Clarion Challenge](#)

Clarion Magazine v1n2 - 3/01/1999
(Mar 8, 1999)

[The Clarion Challenge - A String Parser](#)

Clarion Magazine v1n4 - 5/01/1999
(May 17, 1999)

[The Clarion Challenge - Results](#)

Clarion Magazine v1n3 - 4/01/1999
(Apr 5, 1999)

[The Clarion Magazine Countdown Contest](#)

Clarion Magazine v1n1 - 2/01/1999
(Feb 1, 1999)

[The Clarion Magazine Technology Poll](#)

Clarion Magazine v1n8 - 9/01/1999
(Sep 7, 1999)

[The Clarion Open Source Project](#)

Clarion Magazine v1n1 - 2/01/1999
(Feb 8, 1999)

[The Cranky Programmer](#)

Clarion Magazine v2n1 - 1/01/2000
(Jan 25, 2000)

[The Cranky Programmer - Install THIS!](#)

Clarion Magazine v1n8 - 9/01/1999
(Sep 7, 1999)

[The Cranky Programmer: Nits And Bits](#)

Clarion Magazine v1n11 - 12/01/1999
(Dec 7, 1999)

[The How And Why of WHO, WHAT, and WHERE](#)

Clarion Magazine v1n2 - 3/01/1999
(Mar 29, 1999)

[The Novice's Corner - Designing Databases](#)

Clarion Magazine v1n4 - 5/01/1999
(May 3, 1999)

[The Novice's Corner - Getting A Grip On Clarion](#)

Clarion Magazine v1n1 - 2/01/1999
(Feb 22, 1999)

[The Novice's Corner - Many-To-Many Relationships](#)

[Clarion Magazine v1n5 - 6/01/1999](#)
(Jun 14,1999)

[The Novice's Corner - Understanding Templates And Embeds](#)

Clarion Magazine v1n2 - 3/01/1999
(Mar 19,1999)

[The Novice's Corner: Clarion Code 3](#)

Clarion Magazine v2n1 - 1/01/2000
(Jan 25,2000)

[The Novice's Corner: Understanding Clarion Code](#)

Clarion Magazine v1n11 - 12/01/1999
(Dec 14,1999)

[The Novice's Corner: Understanding Clarion Code](#)

Clarion Magazine v1n9 - 10/01/1999
(Oct 26,1999)

[The Open Source Index For Subscribers](#)

Clarion Magazine v1n2 - 3/01/1999
(Mar 8,1999)

[The Other Way To Use OLE](#)

Clarion Magazine v1n6 - 7/01/1999
(Jul 20,1999)

[The Other Way To Use OLE - Part 2](#)

Clarion Magazine v1n8 - 9/01/1999
(Sep 7,1999)

[The Other Way To Use OLE - Part 3](#)

Clarion Magazine v1n10 - 11/01/1999
(Nov 2,1999)

[The SQL Answer Cowboy](#)

Clarion Magazine v1n6 - 7/01/1999
(Jul 13,1999)

[The SQL Answer Cowboy](#)

Clarion Magazine v2n1 - 1/01/2000
(Jan 4,2000)

[The SQL Answer Cowboy](#)

Clarion Magazine v1n11 - 12/01/1999
(Dec 7,1999)

[The SQL Answer Cowboy](#)

Clarion Magazine v1n7 - 8/01/1999
(Aug 17,1999)

[The SQL Answer Cowboy](#)

Clarion Magazine v1n3 - 4/01/1999
(Apr 6,1999)

[The SQL Answer Cowboy Answers!](#)

Clarion Magazine v1n3 - 4/01/1999
(Apr 26,1999)

[The Unofficial Clarion OOP Page](#)

Clarion Magazine v1n1 - 2/01/1999
(Feb 6,1999)

[Tom Hebenstreit New Reviews Editor](#)

Clarion Magazine v1n7 - 8/01/1999
(Aug 10,1999)

[Tool Talk: New Products At DevCon](#)

Clarion Magazine v1n9 - 10/01/1999
(Oct 14,1999)

[TopSpeed Headed For Java](#)

Clarion Magazine v1n8 - 9/01/1999
(Sep 29,1999)

[TopSpeed's New Direction: The Web](#)

Clarion Magazine v1n8 - 9/01/1999

(Sep 28,1999)

[Understanding OOP Interfaces](#)

Clarion Magazine v2n1 - 1/01/2000
(Jan 18,2000)

[Undocumented Debugging](#)

Clarion Magazine v1n7 - 8/01/1999
(Aug 17,1999)

[Updated COSP](#)

Clarion Magazine v1n7 - 8/01/1999
(Aug 10,1999)

[Updated Debugging/Profiling Classes](#)

Clarion Magazine v1n2 - 3/01/1999
(Mar 29,1999)

[Using MS Word With OLE: The Easiest Way](#)

Clarion Magazine v2n1 - 1/01/2000
(Jan 11,2000)

[WebBuilder Skeleton Basics II: Logos and Fonts](#)

Clarion Magazine v2n1 - 1/01/2000
(Jan 18,2000)

[WebBuilder Skeleton Basics: Which? When?](#)

Clarion Magazine v2n1 - 1/01/2000
(Jan 11,2000)

[Working With Control Files I](#)

Clarion Magazine v1n6 - 7/01/1999
(Jul 27,1999)

[Working With Control Files II](#)

Clarion Magazine v1n7 - 8/01/1999
(Aug 10,1999)

[xBASE Y2K Driver Patches](#)

Clarion Magazine v2n1 - 1/01/2000
(Jan 18,2000)

Copyright © 1999-2000 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca.

published by
CoveComm Inc.

clarion magazine

Good help isn't that hard to find.

\$6.²⁵/month[Main Page](#)[Log In](#)
[Subscribe](#)
[Renewals](#)[Frequently Asked
Questions](#)[Site Index](#)
[Article Index](#)
[Author Index](#)
[Links To
Other Sites](#)[Downloads](#)
[Open Source
Project](#)
[Issues in
PDF Format](#)
[Free Software](#)[Advertising](#)[Contact Us](#)

Feature Article

January, 2000

Using MS Word With OLE: The Easiest Way!

by **George Petrov**

I have always looked for ways to control Microsoft Word, partly because doing reports in Clarion has always been a major pain. End-users always welcome more power, almost everybody knows Word and many people work with it often. So when your end users ask the question "Can we integrate with Word?" you don't have to get a heart attack any more.

If you think that I will present you a product review of QuickDDE you are wrong! Instead I will show you how you can do it all by yourself, maybe by just reusing some of the classes I wrote for this article.

There are many different ways to automate Word, and I've been experimenting with all of them. Here is a list:

- **Using standard DDE:** This is the oldest way to control Word. You can send some few predefined commands to Word. It's very limited, and very slow, and you will have much trouble synchronizing your commands.
- **Using the standard Clarion OLE:** This is also a pain. Most of the time you want Word to be started as a separate application, not integrated in your window. Unfortunately Clarion OLE is window-based. You have to fight with the terrible property syntax for accessing OLE methods and properties. Also all the commands you can send are dependent on the International language Windows is using – not everybody is using the US version of Microsoft Word! At least with this method of accessing Word you have the power to control it through all the Visual Basic commands, but it is slow.
- **OLE Class Generator:** With this product can generate wrap classes to let you access Word in a very easy and intuitive way, just as you will in Visual Basic! It takes away the pain of dealing with OLE on your Window and you don't need the property access syntax to let the things work: you just call Clarion methods. However, the Word classes can be so big that sometimes the OLE Class Generator can have trouble generating them, and even if it succeeds you won't be able to compile them with Clarion. So a major cleanup will be needed; you will have to leave only the methods in the classes you really need and throw the rest away.
- **QuickDDE:** I have to mention this product, because this is the first tool I wrote that connects to Word. In the 2.x version this tool uses OLE for automating Word. It does that by a combination of the two solutions above: it has a wrapper class that does all the work with Word. However this class still uses standard Clarion OLE to get things done. This is slow because the standard Clarion OLE uses late binding.

[Using MS Word With OLE:
The Easiest Way](#)
(Jan 11,2000)[WebBuilder Skeleton
Basics: Which? When?](#)
(Jan 11,2000)**TopSpeed****Clarion 5**
by TopSpeed

**"When
you need
the best"**

**UltraTree
Platinum**

[Click Here](#)

The Keys To The Kingdom

Recently I was reading the marvelous articles in Clarion Magazine written by [Jim Kane about OLE](#). Well, I have to say I read them with my mouth open. This was really all I needed to get OLE running from within Clarion! He presented me indeed the keys to the OLE Kingdom.

I always had trouble with the standard Clarion OLE implementation. It is too clumsy and slow, and very often it just doesn't work. Also it is always connected to a visual OLE control on the window, and that is not always desired, such as when you are using just OLE Automation (more on that later).

Having this great knowledge I was able to build a class that allowed me to access Word very easily. This class uses the early binding method for accessing OLE, which guarantees very fast access to Word and a lot of power. Using this class you may feel like Aladdin releasing the genie from the lamp!

The class I will demonstrate here is not by any means complete; you can reuse and extend it to suit your own needs. I will just try to give you a good start.

What Is OLE Automation?

OLE Automation was originally developed as a way for applications (such as Word and Excel) to expose their functionality to other applications, including scripting languages. The intention was to provide a simple way to access properties and call methods that put as little strain as possible on the automation client, which in this case is your Clarion application.

Automation works absolutely the same as OCX (or OLE – it's the same) controls. The only difference is the automation controls (i.e. servers) do not have their interfaces embedded into your application's window.

For calling OLE methods there are in general two options – early binding and late binding. Binding refers to the kind of connection between the OLE object and the client (application) which is using the object.


Late Binding

In the early days of OLE only late binding was possible. This happens through a wrapper interface called `IDispatch`. This interface looks up the name of the method you want to call on the OLE object, translates all the parameters you pass, and eventually makes the call. There is quite an overhead incurred because all the method translation is done at runtime, as well all syntax checks. You will definitely get a headache if you don't get things running immediately.

Clarion's standard OLE builds an extra layer upon the `IDispatch` interface to translate the Clarion calls to `IDispatch` calls. This slows down the access even more.

Early Binding

After a while Microsoft saw that all those `IDispatch` translation calls were actually very slow. So they introduced direct `VTable` calls. As you may recall from Jim's article a `Vtable` is just an array of pointers to the methods an interface contains. But because calling a `Vtable` method and a method through `IDispatch` is very different, Microsoft decided to create a merge of `VTable` calls and `IDispatch` interfaces called *dual* interfaces. Those interfaces are capable of supporting both early and late binding, i.e. both `VTable` calls or `IDispatch` calls. Early binding calls OLE methods directly using the pointer to the method you want. That makes them the fastest way to access OLE methods. Used this way OLE performance is equal to that of calling DLL functions: no extra overhead is involved.



Your source for
development tools
and add-ons.

Your outlet for
application sales.



May 23-27, 2000
May 23-27, 2000

Register Today!

In this article I will use early binding. This way I will have the fastest access and also a compile-time check of all calls.

NOTE: There is a little catch here. Only Word 97 and later versions support early binding. Do *not* try the following method on Word '95 or earlier versions.

Let's Rock & Roll

So enough general talking! Here's a list of what I'm going to do:

1. Use Jim Kane's OLE classes to connect to Word
2. Derive those classes and extend them with some Word methods
3. Have lots of coffee

First I recommend that you do a triple reading of [Jim Kane's articles](#). Diving into Jim's code is also helpful from time to time, but this is only if you want really to understand how things work. I suppose 99.5% of you don't want that and just want to reuse the class that I will build here – well that's OK!

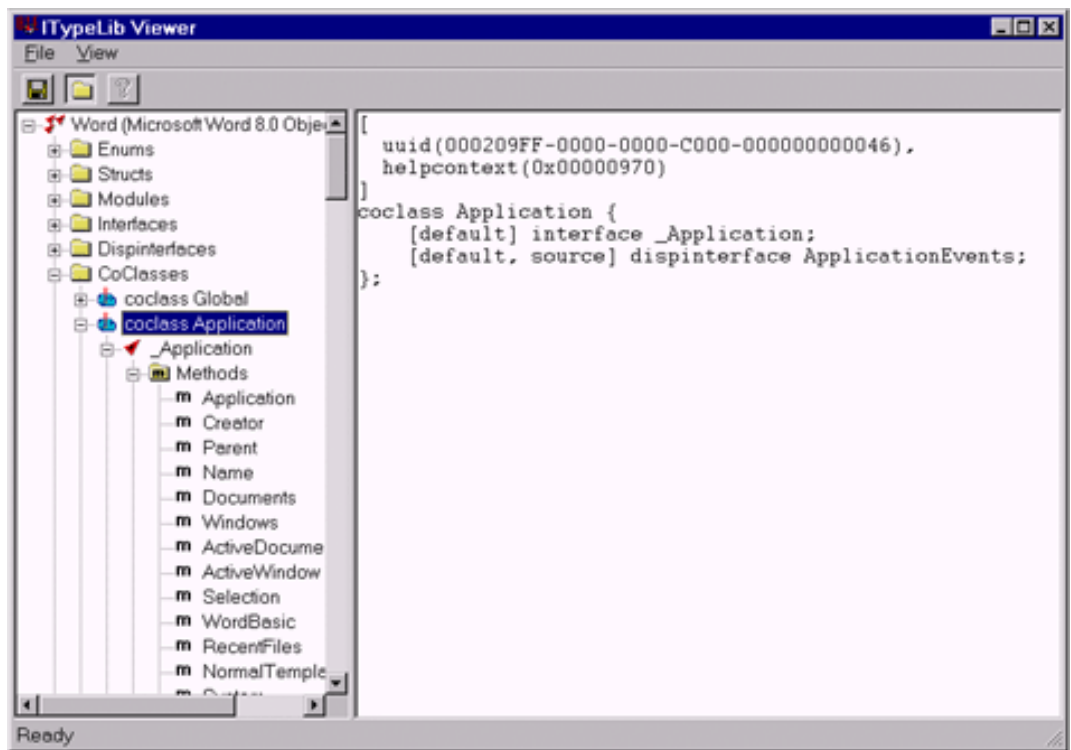
Getting Started

To start I need the [source of Jim Kane's classes](#) contained in the following files:

- *HoleCl.clw and HOleCl.inc* The base class `hOleClType` that I will base my class on.
- *OleCl.clw and OleCl.inc* The `OleClType` class that I will need to initialize OLE.
- *StrCl.clw and StrCl.inc* A utility class for dealing with OLE string types.
- *ICall.a* The low-level assembler routine to make the actual calls to the OLE methods.
- *VarCl.clw and VarCl.inc* The very simple Variant class that I wrote. I will discuss that in another article. It handles the Variant data type, used very often in OLE calls
- *OLE32.LIB* Library with declarations of some OLE API functions that Clarion is missing.

Next I will need the OLE View tool provided by Microsoft (you can find it on their web site <http://www.microsoft.com/com/resources/oleview.asp>)

Figure 1. The OLE View Tool.



Exploring The Word Type Library

Using the OLE view tool you can read the type library (the description of all the methods and properties) that Word offers to be called.

Start OLE View and choose File/View TypeLib. Then go to your Office directory and choose MSWORD8.OLB for Word 97 or MSWORD9.OLB for Word 2000.

As you can see in the above screen there are quite a lot of things in a Type Library.

Start at the CoClasses. This is the main entry for the Automation - Application CoClass; when you expand it you will see all the methods that you can call under the _Automation interface. Take a look at the Name method; I will use this in my first test app to get the name of Word.

For my test app I will need quite a lot of IDs that you can find here, and I will explain how to get those in a minute.

Building My Own Word Class

I will make a new project first. This will be a handcoded project *not* an app. So choose Project|New and call the project TestWord.prj

Set the project to 32 bits and the main file to TestWord.clw. Add to the External Source section of the project all the needed CLW files as listed in the section above. Also add the ICall.a file. Add the OLE32.LIB file to the Library section.

Figure 2. Main file TestWord.clw.

```

PROGRAM
  MAP  END
  ! These equates needed only because I'm using a project
  ! instead of app in the generator and I want to
  ! compile my ABC compatible classes without problems
  _ABCDllMode_  EQUATE(0)
  _ABCLinkMode_ EQUATE(1)

  INCLUDE('OLECL.INC')
  INCLUDE('MSWORD8.INC')
OLECl      OLECLType
MyWord     WordClass
  CODE
  OLECl.InitOLE(CoInit_ApartmentThreaded)
  !0=no error messages, 1=msg on error only, 2=verbose
  MyWord.Init(1)

  MyWord.SetVisible(True)
  MESSAGE('MyWord.Name = ' & MyWord.GetName())

  MyWord.Kill()
  OLECl.killOLE()

```

As you can there see nothing to it! First I declare two variables (objects), one for the OLEStartCl and one for my WordClass. Then I initialize OLE, initialize my WordClass, make Word visible, show its name and shut down. You would *love* to use Word in this way, wouldn't you?!

But now I have to build the classes, of course.

I will start with the include file.

Figure 3. The include file MSWORD8.INC.

```

!ABCIncludeFile

OMIT('_EndOfInclude_',_CWordClassesPresent_)
_CWordClassesPresent_ EQUATE(1)

!Other Classes
  Include('HoleCl.Inc')
  Include('StrCl.inc')

WordClass CLASS(hOleClType),TYPE,MODULE('MSWORD8.CLW'),←
  LINK('MSWORD8.CLW',_ABCLinkMode_),DLL(_ABCDllMode_)
!--- Methods ---
Init      PROCEDURE(BYTE pDebugMode=0),BYTE,PROC
GetName   PROCEDURE(),STRING
SetVisible PROCEDURE(BYTE OnOff)
          END
_EndOfInclude_

```

As you can see I'm deriving a new class from hOleClType, which is the base OLE class. I'm

also overriding the Init method and have some extra methods.

Next I'm going to implement the new class.

Figure 4. Implementation file MSWORD8.CLW.

```
MEMBER( )

_ABCD11Mode_  EQUATE(0)
_ABCLinkMode_ EQUATE(1)

        INCLUDE('MSWORD8.INC')
! This is the include file for the
! variant class, I will discuss
! this one in another article
        INCLUDE('VARCL.INC')
! Needed API calls prototypes
! I will describe those later
MAP
        MODULE('Windows')
                SysFreeString(LONG),LONG,PASCAL,PROC
                OleRun(LONG),LONG,PASCAL,PROC on
        END
END

!Global data for this module
Return:benign equate(0)
Return:Fatal  equate(3)
Return:Notify equate(5)
```

This was the easy part because it is always the same. But now comes the more difficult part.

[Read Part 2](#)

[Using MS Word With OLE: The Easiest Way](#)

(Jan 11,2000)

[WebBuilder Skeleton Basics: Which? When?](#)

(Jan 11,2000)

Copyright © 1999-2000 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca.

[Main Page](#)

[Log In](#)
[Subscribe](#)
[Renewals](#)

[Frequently Asked Questions](#)

[Site Index](#)
[Article Index](#)
[Author Index](#)
[Links To Other Sites](#)

[Downloads](#)
[Open Source Project](#)
[Issues in PDF Format](#)
[Free Software](#)

[Advertising](#)

[Contact Us](#)

Feature Article

January, 2000

Using MS Word With OLE: The Easiest Way!

by **George Petrov**

[Read Part 1](#)

As explained in Jim's articles, to access an OLE object you need to do the following:

1. Declare interface equates
2. Get the CLSID of the main object
3. Get the IID of the interface you want to access
4. Get the IDs of the methods you want to call
5. Construct the Init method, listing all the interfaces and methods you want to call
6. Implement the extra methods you want to have in your class.

With a little more digging with OLE View the above should be easy to find.

Declaring Interface Equates

The base class requires that for each interface used an equate declaration is needed to identify it in your code. So let's do it. For now I will use two interfaces: the main IWordObject and the secondary IWordApplication.

Figure 5. Equates for Interfaces.

```
!Equate for Interfaces
!1-19 is reserved for holeCl interfaces
    ITEMIZE(20)
    !List all interfaces called here
IWordObject      EQUATE
IWordApplication EQUATE
    END
```

If you get more interfaces later on just add them to the list. Position is not important.

Getting CLSID

[Using MS Word With OLE:
The Easiest Way](#)
(Jan 11,2000)

[WebBuilder Skeleton
Basics: Which? When?](#)
(Jan 11,2000)

TopSpeed

Clarion 5
by TopSpeed

FREE Microsoft
Internet
Explorer

etc
EVENT SPONSOR



The CLSID I'm looking for is just the UUID of the CoClass displayed in OLE View. When you choose the Application CoClass, on the right side you will see its UUID. Select and copy it.

I'm going to do some work on translating that to a Clarion group. Just initialize the group with hex values.

Figure 6. CLSID of Word.Application.

```
! IWordApplicationObject
!  uuid(000209FF-0000-0000-C000-000000000046) ,
CLSID_WordObject  GROUP
data1             ULONG(000209FFH)
data2             USHORT(0000H)
data3             USHORT(0000H)
data41           BYTE(0C0H)
data42           BYTE(000H)
data43           BYTE(000H)
data44           BYTE(000H)
data45           BYTE(000H)
data46           BYTE(000H)
data47           BYTE(000H)
data48           BYTE(046H)
END
```

As you can see I've copied the UUID in the comment first and then put the right hex values in the structure (I always copy this structure and then change the values only).

Getting IID

The IID of the interface I'm looking for is just the UUID of the interface under the CoClass in OLE View. In my case I need _Application. So I choose the Application CoClass and below it _Application interface. On the right site you will see the UUID of it. Select and copy it.

Figure 7. IID of the _Application interface.

```
! IWordApplication
!  uuid(00020970-0000-0000-C000-000000000046) ,
IID_WordApplication  GROUP
data1             ULONG(00020970H)
data2             USHORT(0000H)
data3             USHORT(0000H)
data41           BYTE(0C0H)
data42           BYTE(000H)
data43           BYTE(000H)
data44           BYTE(000H)
data45           BYTE(000H)
data46           BYTE(000H)
data47           BYTE(000H)
data48           BYTE(046H)
END
```



Getting IDs Of The Methods

Next I'll build a table with the IDs of all methods I'm going to call. This can get very nasty; quite a lot counting is involved, actually. You have to know that every method is actually a four byte pointer. So getting down in the line you need to count every method by four starting with zero for the first.

In my case I need the Name method from the `_Application` interface. You would think that is easy; it's the fourth method so $4 * 4 - 4 = 12$; and the address will be 12, right?

Well, it is not that easy. Remember, as Jim wrote, that Microsoft is involved here. As you can see in OLE view, after all the methods for a given interface there is an option called *Inherited Interfaces* (you may need to scroll a little). You can see that interfaces get derived. So calculating the method offset involves also counting the methods of all inherited interfaces!

When you open this level of the tree you will see again `_Application`. This is the same interface as you were looking at so skip it. Go down under it and choose its option for Inherited Interfaces.

Now you will see that the `_Application` interface is derived from `IDispatch`! Very nice, you say! Now open the `IDispatch` and you will see that it also contains four methods. Okay, you have four now. Open the Inherited interfaces of `IDispatch` and you will see that it is `IUnknown`! So count this one too – it contains three methods.

Finally that's it! I have seven extra methods to count! Seven methods multiplied by 4 bytes is 28 bytes to add.

This means that the calculation for the offset of the Name method will be:

$$4 * 4 - 4 + 28 = 40$$

To make things easy I will add always 24 bytes ($28 - 4$). I also always add the interface equate that I've declared earlier and multiply it by 100H (hex).

Ok enough math, lets see what the method table looks like:

Figure 8. List of called methods.

```
!Vtable Offset - List all methods called
IWordApplication_Name          ←
      EQUATE(IWordApplication*100H + 4*4 + 24)
IWordApplication_Visible      ←
      EQUATE(IWordApplication*100H + 28*4 + 24)
```

As you can see in the code above, I also added another method called `Visible`. I will need this method to unhide Word after starting it. However there is also something special to say about the offset number of this method.

The first thing about the `Visible` method is that to find its offset you have to begin counting from the first method till you find the `Visible` method. But wait a minute, you may say, there are two `Visible` methods! How can that be? Actually the `Visible` method is a property, although it is displayed as a method. Because properties can be set or retrieved, they are implemented by two methods – one for the Set operation and one for the Get operation. But how do you see the difference if they have the same name? Look at their declaration on the right side. You will see that the one method returns a value, while the other takes a value as a parameter. In my case I want to set the value so I use the second one.

Constructing The Init Method

Now I can finally get into the implementation of the methods. I will start with the Init method.

A normal flow of the Init method will be:

1. Use the AddIID method to link the interface equate with the address of the IID structure
2. Use the AddMethodName to link the method name equate with a logical name (used for better error reporting and debug)
3. Call CoCreate with the address of the main object and the Interface you want

It should look like this:

Figure 9. The Init method

```
WordClass.Init    PROCEDURE(byte pDebugMode=0)
Res BYTE,AUTO
CODE
CLEAR( SELF)
res = Parent.Init(pDebugMode)
IF ~Res THEN
    SELF.AddIID( IWordApplication, ←
        ADDRESS( IID_WordApplication), 'IWordApplication')

    SELF.AddMethodName( IWordApplication_Name, ←
        'IWordApplication_Name')
    SELF.AddMethodName( IWordApplication_Visible, ←
        'IWordApplication_Visible')

    IF SELF.CoCreate( ADDRESS( CLSID_WordObject), ←
        IWordApplication) THEN RETURN( Return:Fatal).
END
RETURN( Res)
```

Unfortunately it seems that this code works only for OLE in-process servers (i.e. that are running as DLL or OCX within the address space of your application). I'm dealing with an out-of process server now because Word runs in a separate address space.

The Headache

This was the biggest headache I ever have had. I was so close to getting it to work and the above code just refused to cooperate. Word didn't start as it should, it just returned an error each time that the interface `_Application` does not exist! This is not possible, of course, because I definitely saw it in OLE View under the `Application CoClass`.

It cost me days of digging in all the OLE documentation, looking at the C++ samples of implementing Automation with Office, diving into lots of source...it was terrible.

After lots of experimenting I saw that the only interface I could get from the `CLSID Application` was `IUnknown`! Strange, not even `IDispatch` – nothing!

I continued inspecting the entire MFC C++ classes source that Microsoft uses in Visual C++. After lots of sleepless nights I finally found it! It was a very hidden OLE call named `OLERun`!

This call starts all things up – so it starts Word! And it requires as a parameter the pointer to

the IUnknown interface. Well, I had that already. The rest was getting a pointer to my real interface `_Application` from the IUnknown after the `OleRun` call.

I know it sounds complicated but this is the Microsoft way – its never easy.

So my `Init` code now looked like this:

Figure 10. The working Init method.

```

WordClass.Init      PROCEDURE(byte pDebugMode=0) !,byte,proc
Res BYTE,AUTO
pVtable long
CODE
CLEAR( SELF)
res = Parent.Init(pDebugMode)
IF ~Res THEN
  SELF.AddIID( IUnknown,ADDRESS( IID_IUnknown), 'IUnknown' )
  SELF.AddIID( IWordApplication, ←
    ADDRESS( IID_WordApplication), 'IWordApplication' )
  SELF.AddMethodName( IWordApplication_Name, ←
    'IWordApplication_Name' )
  SELF.AddMethodName( IWordApplication_Visible, ←
    'IWordApplication_Visible' )

  IF SELF.CoCreate(ADDRESS( CLSID_WordObject), ←
    IUnknown) THEN RETURN(Return:Fatal).
  pVtable = SELF.GetVtable(IUnknown)
  OleRun(pVtable)
  !Get IWordApplication with QueryInterface from IUnknown
  If SELF.QueryInterface(IUnknown, ←
    Address( IID_WordApplication), IWordApplication ) ←
    then Return(Return:Fatal).

END
RETURN(Res)

```

Since I'm using a new interface here (IUnknown) I need to declare that too.

First add it to the interface equates table:

Figure 11. Equates for Interfaces.

```

!Equate for Interfaces
!1-19 is reserved for hOleCl interfaces
  ITEMIZE(20)
  !List all interfaces called here
IUnknown      EQUATE
IWordObject   EQUATE
IWordApplication EQUATE
END

```

Then it has to have its own IID structure too:

Figure 12. IID of the IUnknown interface.

```

! IUnknown
! uuid(00000000-0000-0000-C000-000000000046),
IID_IUnknown    GROUP
data1           ULONG(00000000H)
data2           USHORT(0000H)
data3           USHORT(0000H)
data41          BYTE(0C0H)
data42          BYTE(000H)
data43          BYTE(000H)
data44          BYTE(000H)
data45          BYTE(000H)
data46          BYTE(000H)
data47          BYTE(000H)
data48          BYTE(046H)
END

```

Yes! I finally have a working `Init` method! This was the big headache but now its working and I'm very happy. After this the implementing of the other methods is just a breeze.

Constructing The Class Methods

The methods `GetName` and `SetVisible` need to be implemented:

Figure 13. Other class methods.

```

WordClass.GetName      PROCEDURE( )
ReturnValue            STRING(512),AUTO
lpBstr                LONG(0)
CODE
  IF ~SELF.ICall(1, IWordApplication, ←
      IWordApplication_Name,ADDRESS(lpBstr))
  IF SELF.StrCl.BSTRToCWStr(lpBstr,ReturnValue) THEN
    CLEAR(ReturnValue)
  END
END
IF lpBstr THEN SysFreeString(lpBstr) .
RETURN(ReturnValue)

WordClass.SetVisible   PROCEDURE(BYTE OnOff)
Var                    VariantCl
CODE
  SELF.ICall(1, IWordApplication, ←
      IWordApplication_Visible,Var.Init(OnOff,VT_BOOL))

```

The GetName Method

There is something special in the `GetName` method: it returns a `BSTR`, or `BString`. This is an OLE type that has to be converted to a Clarion string. Jim wrote a very nice class to do this and integrated it in his OLE class. So to use it first I will pass just the address of a `Long` that will receive the pointer to the `BSTR`. Then I convert the `BSTR` to a Clarion string. Last but not least, I'll make sure there was indeed a pointer to a `BSTR` returned, in which case I'll have to call an API call to free it. Note that you have to free all the variables that are allocated by the OLE methods in this way.

The SetVisible Method

Of course the `SetVisible` method has something special too: it requires a parameter of the type `VARIANT_BOOL`. I won't go here in details, but I will tell you that I wrote a small class to handle `Variants`. To use it just declare a variable of type `VariantCl` and then where it is required call the method `Var.Init(Value, Type)`. That's all!

I will discuss the `Variant` class in detail in another article.

The Grand Finale

This is the Grand Finale. If you followed along, you've done it! You can control Word via OLE now! And I mean really control it – you have all the power. There are no more constraints like those imposed by TopSpeed's standard OLE. You possess the keys to the Kingdom!

But George, you might say, you've done all that work just to get Word started! Aren't you exaggerating a little? I could do the same with a simple call to `ShellExecute`! Yes, I know, but I'm just getting started! Starting is always the most difficult step, but when it's already going, there is no stopping! So be patient! I will show you more and more each time! This is just a warm-up!

Coming Up Next Time

In the next article I will implement more useful features, including:

- Word collections: how to get them, iterate through them and use them
- Other Word Objects: how the Document object allows to do more with the Word document
- The Variant class: how it works and how to use it

If you have your own ideas and questions feel free to email me. I hope you enjoy reading this article, and I'll see you next time.

[Download the source code](#)

George Petrov was originally born in Sofia, Bulgaria. He is now 28 years old and for the past 10 years has lived in the Netherlands, where he graduated in Computer Science. He worked for a few years with Philips Communication & Processing, on old mainframes. But he did survive and even ported a Zip/Unzip compatible program to those crazy operating systems. But his passion was always programming the PC, which was his greatest hobby. For the last four years he has worked as a Software Engineer at [Princen IT](#) where he specializes in Clarion tools and technology and leads technical researches and consulting.

[Using MS Word With OLE: The Easiest Way](#)

(Jan 11,2000)

[WebBuilder Skeleton Basics: Which? When?](#)

(Jan 11,2000)

Copyright © 1999-2000 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca.

published by

CoveComm Inc.

clarion magazine

Good help isn't that hard to find.**\$6.²⁵/month**[Main Page](#)[Log In](#)
[Subscribe](#)
[Renewals](#)[Frequently Asked
Questions](#)[Site Index](#)
[Article Index](#)
[Author Index](#)
[Links To
Other Sites](#)[Downloads](#)
[Open Source
Project](#)
[Issues in
PDF Format](#)
[Free Software](#)[Advertising](#)[Contact Us](#)**Feature Article**

January, 2000

Web App Skeleton Basics: Which? When?

by Steve Parker

WebStyles or Skeletons, depending on who you talk to - and the time of day - are at once extraordinarily simple and devilishly complicated

They are simple because, in the target market, the Clarion developer does not need to concern her/himself with them or, indeed, with HTML to any significant degree. The Clarion developer creates a data driven application that works and turns it over to the customer. The customer provides an HTML expert (a/k/a web-head) who customizes the visual elements.

They are simple because WebBuilder almost entirely separates the presentation layer from the function layer. I can place buttons to the left or right of another control, above or below another control and that is (virtually) the sum total of what I need to be concerned with visually.

They are difficult because WebBuilder almost entirely separates the presentation layer from the function layer. I can place buttons to the left or right of another control, above or below another control and that is (virtually) the sum total of what I can do (do I hear an echo?). However, these controls may not appear quite where I expect them to. I am accustomed to much greater control than this.

They are difficult because many Clarion developers are responsible to their customers for the complete app, both form and function. This means learning HTML. But the real difficulty is that many things we are used to controlling in code (usually Clarion) are no longer so controlled. And when we do locate the appropriate embed, the method of control is radically different. Control is via external files, not true embeds.

Compounding the problem is that WebBuilder is in beta and therefore subject to change. Given some recent postings from the Development Centre, some of the changes that may be coming may well be significant. Likewise, the documentation we need cannot be assembled until all changes are complete; documentation is the very last part of the product finalized. Yet we are beginning to work with this technology. Catch-22.

First Things First

[Using MS Word With OLE:](#)[The Easiest Way](#)
(Jan 11,2000)[WebBuilder Skeleton
Basics: Which? When?](#)
(Jan 11,2000)[Skeleton Basics
Part 1](#)[Skeleton Basics
Part 2](#)[Skeleton Basics
Part 3](#)[Skeleton Basics
Part 4](#)

In the most general terms, a single skeleton governs the generation of HTML for a single Windows control. However, a few skeletons chain so that more than one skeleton is responsible for the production of a segment of HTML. At least one skeleton generates multiple HTML controls. The question is "Which skeleton controls which HTML creation? And how do I find out which?"

Before you can customize, you need to know what is to be customized.

While waiting for the documentation, there are a few ways to discover what you need to know.

Obviously, you can open each skeleton file in a text editor. If you are comfortable, really comfortable, with HTML (I, for one, am not), reading the HTML will tell you what the skeleton does.

Alternately, you can open each skeleton in a visual HTML editor and see what the display looks like. Unfortunately, several of the skeletons display nothing and much of the most important stuff is not displayed.

If you are willing to sacrifice any chance whatsoever of election to the Lazy Programmers Society, you can do both: use a visual tool as well as a text tool. For example, you can open a skeleton in a browser and use the source viewing option to check the actual code. A browser and WordPad ... a real developer's combination!

As I happen to use this technique and consider myself a paid up member of LPS (well, if I ever got motivated enough to send in my dues...), I just have to believe there is a way to mitigate this stigma.

Actually, there are two.

First, many of the skeleton files are so well named that what they do is perfectly clear. For example, Box, Button, Check, Entry, Group, Item, Menu, Panel, Prompt, Radio, Region, Spin and Splash HTML files very, very clearly indicate the control generated. Others are only a little bit less expressive, like Select (an HTML "select"). String.HTM will trick you; it contains no string but cascades to another skeleton.

For these, a quick look in a browser is usually enough.

In fact, only a few skeletons actually need to be examined to discover their purpose or method of operation.

There is a second way of determining which skeleton is invoked and when it is called. This way is quite useful when you cannot otherwise determine which skeleton was called (this will happen to you). Also when more than one skeleton might be called – like the two for sheets or tabs – and you can't readily figure out which.

If you check the generated HTML of a running app, using your browser's source viewing option, you will see HTML comments indicating the beginning and the end of HTML generated by most of the skeletons.

For example, the splash screen for Invoice.app contains:

```
<!-- Panel.htm -- Start -->
```

```
<head><meta name="ts-control" content="panel">
```

and

```
<!-- Image.htm -- Start -->
```

```
<a><img src='/SSANTHUR.GIF' width=152 height=166></a> <!-- Image.htm  
-- End -->
```

One powerful way to use this information is to copy the HTML out of a running app. Paste it into the code view of a visual editor. As you manipulate this static

HTML, it becomes very easy to discover which skeleton generated the HTML you've been working with. Not only that, you will discover what HTML is generated (there is much not contributed by the skeletons but by the HTML parser) and what can be done with it.

The Exceptions

Window and WinCore.HTM do not generate comments, which presents a problem. Table.HTM controls multiple page elements. String.HTM does not contain any text data. These are the major exceptions.

Window.HTM, as it turns out, only provides the HTML tags which frame all the other HTML. It doesn't generate any runtime HTML.

WinCore provides the basic structure for all HTML pages. WinCore is where you specify site-global background colors and/or images.

However, these two skeletons, Window.HTM and WinCore.HTM, are quite important.

Any images or colors setup in WinCore will affect every app using this skeleton set and every page in those apps. But what if you want one page or set of pages to have a different background color or image? Can't be done; every single page is created out of WinCore.

"Ah ha!," you say, "there are template prompts to override the defaults. I'll just nominate a different WinCore.HTM."

Can't be done. WinCore is the one skeleton that cannot be conveniently overridden.

But there is a way. To accomplish this effect, you need to nominate a different Window.HTM which, in turn, calls a different WinCore.HTM, shown in Figure 1.

Figure 1. Using a different Window.HTM.

```
<!-- shpWindow.htm -- Start -->
<html>
<head>
<meta name="ts-control" content="window,application">
</head>
<TSINCLUDE name="colora.htm">
<TSINCLUDE name="SHPwincore.htm">
<TSINCLUDE name="colorb.htm">
</html>
<!-- shpWindow.htm -- End -->
```

Note that I have not only renamed the skeleton but also reflected this change in the comments. Strictly speaking, this is not necessary, but should you need to back trace your generated HTML this will be a major timesaver

"Ok," you say, "that's fine and well. And, it's valuable information, I suppose. But whyever would I want to do this? Isn't this just an academic question?" Perhaps you want a different look on browses and forms. (If you check out my FAQ app at www.par2.com/cws/c5launch.dll/faqs/coolfaqs.exe.0, you will note that the FAQ articles are less readable than they could be or, at least, I think this is the case. And, I think this is due to the common background.)

A Little Knowledge Is A Dangerous Thing: Let's Modify A Skeleton

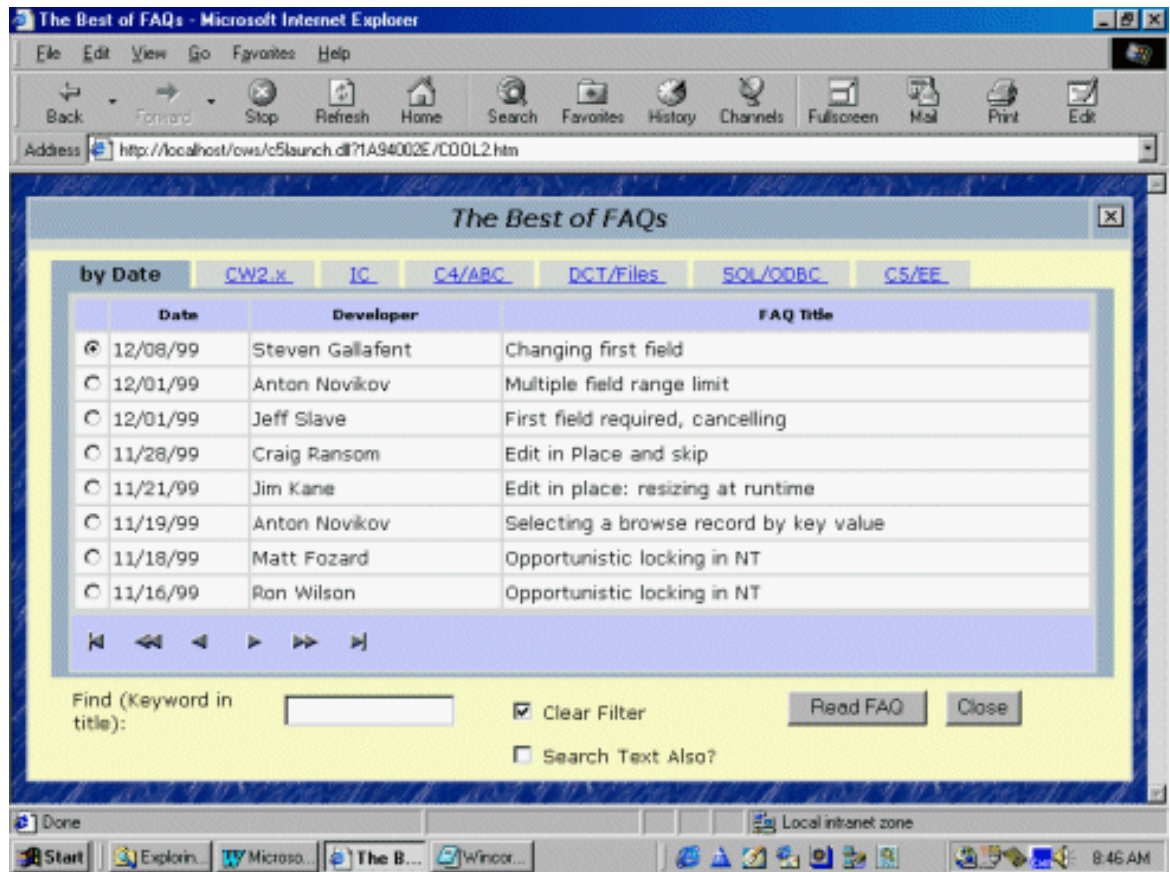
If you have a working familiarity with HTML, there is yet another way of

determining what skeletons control what HTML. This is the highly systematic, ancient and honorable programming technique of trial and error (and how I discovered that WinCore, not Window.HTM, despite the <body> tag, controls backgrounds).

This same trial and error with WinCore.HTM demonstrated that adding the code `background="/plastic.jpg"`

attribute to WinCore.HTM's <body> tag at first seemed Ok but did not give the result expected when the final app was running, as shown in Figure 2.

Figure 2. The FAQs browse with no background on the browse.



(Note the use of a ... ah, very clearly observed image. This makes the result very easy to see.)

Investigating further demonstrates that there is a <table> with a series of four rows (<tr> tags) and five cells (<td> tags) here. If you add the background attribute to these, then you will successively approach the result you want.

Why?-Because the areas for the menus, toolbars and other controls (like browse boxes and entry fields) are empty when working with the skeleton. At runtime, similar <table> tags are created for those controls.

Alternately, you can remove one or more of these rows if you do not need them.

You do need to look closely at the default HTML. And when you do, you will see that the first two cells on the first row contain the page title and the "extra" close button.

If you do not want one or both of these, you can delete or comment out either of the cells or even the entire row (commenting is probably the better course). If you want the title centered, add the center attribute to the cell. Finally, an image placed inside one of these cells will appear at the very top of each page. And,

clearly, you can finagle this location to provide various logo placements.

For "safety" sake, comment your discoveries, as shown in Figure 3, so you can find them later.

Figure 3. Commenting skeleton discoveries.

```
<td width=100% align="center"><font face="verdana"><b><!-- SHP status
text-->

<!-- SHP image here appears at top of page -->

<TSSCRIPT value=Title>

Page Title

</TSSCRIPT>
```

Summary

Not very exciting stuff but one must walk before running.

To modify the look and feel of a Web Builder app, you don't really have options: you must know the (skeletal) source of your HTML. I've presented several ways of determining which skeletons are being used where. Next week I'll start looking at the most common skeleton modifications.

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

[Using MS Word With OLE: The Easiest Way](#)
(Jan 11,2000)

[WebBuilder Skeleton Basics: Which? When?](#)
(Jan 11,2000)

Copyright © 1999-2000 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca.

published by
CoveComm Inc.

clarion magazine

Good help isn't that hard to find.**\$6.²⁵/month**[Main Page](#)[Log In](#)
[Subscribe](#)
[Renewals](#)[Frequently Asked
Questions](#)[Site Index](#)
[Article Index](#)
[Author Index](#)
[Links To
Other Sites](#)[Downloads](#)
[Open Source
Project](#)
[Issues in
PDF Format](#)
[Free Software](#)[Advertising](#)[Contact Us](#)**Feature Article**

January, 2000

Interfaces: Helping Objects To Lie

by David Bayliss

And he that betrayed him had given them a token, saying, Whomsoever I shall kiss, that same is he; take him, and lead him away safely. And as soon as he was come, he goeth straightway to him, and saith, Master, master; and kissed him. And they laid their hands on him, and took him. (Mt 14:44-46)

The text above covers what is probably one of the most famous (or infamous) cases of treachery ever recorded: the betrayal of the Son of God by one of his disciples. The religious issues are there for anyone who chooses to read the account, but I want to look at some of the interfacing issues that the passage brings out.

There are essentially three parties involved in the transaction: Judas, the Lord, and the soldiers.

Now Judas was presented with a problem. His task was to locate and identify the Lord on a dark evening, in an unlit area with plenty of hiding places. Further Judas would know that at least eleven men would be around, at least some of whom were armed and would be prepared to defend their leader with their lives. In order to get to the Lord he was going to have to behave in a style or manner that made him appear as a disciple.

However, Judas didn't want to behave like a disciple, he wanted to betray the Lord to the soldiers. On a fine clear day on an open field that would simply be done using a "Look, he's over there" mechanism. But Judas had to find a way of appearing as a disciple and a traitor *at the same time*.

Judas' solution was clever (even if revolting); put in object-oriented programming terms, he *declared* a subset of the disciple interface (the kiss) to act as the event for the traitor interface (the "pointing out"). The rest is history.

Masquerading Objects

It turns out then when writing object oriented programs you often find yourself in the same predicament as Judas: your want to write one contained object that appears as two

[Understanding OOP
Interfaces](#)
(Jan 18,2000)[WebBuilder Skeleton
Basics II: Logos and Fonts](#)
(Jan 18,2000)[xBASE Y2K Driver Patches](#)
(Jan 18,2000)[Advertise in Clarion
Magazine!](#)
(Jan 18,2000)



different things to two different containers.

Imagine that you are writing a control template to build up a list of customers that are having invoices printed out (this list is to act as an index, perhaps). If you think about embeds for a moment you will probably agree that you need something in the `TakeRecord` embed (to read the customer names when the record is fetched) and something in the `TakeEvent` embed (for user interaction with the window).

Now being a good little OOPer you will know that rather than splurging that code throughout a template it is far better to simply derive and put all the code in one place. So you look up `TakeRecord`, derive the process class and then hunt around for the `TakeEvent` method. Ouch, it isn't there. Ok, forget deriving the process class. Derive the `WindowManager` instead, override the `TakeEvent` method and then the `TakeRecord` method...ugh! It isn't there!

So what now? Probably a diatribe on the Web under the general heading of OOP stinks! But it is quite important to see *why* you get the smell in this case. Your task (a window display of data provided by a process) is trying to straddle two completely different areas of functionality: user interface and data collection. If the system *had* let you bundle code into one object, just think what a *mess* that code would have been. Worse yet, suppose the data processing had been complex (not just collecting user names) and you then had a requirement to mail-merge it into an e-mail. How pleased would you be trying un-pick all the data processing code from the list-box management code you had written?

Of course I'm depressing you to prepare you for the solution: Interfaces. I'll look at the details in a moment but first I'll follow along on the example.

In C55 there are two interfaces (amongst others) defined,

```
RecordProcessor INTERFACE
TakeRecord          PROCEDURE , BYTE , VIRTUAL , PROC
TakeClose           PROCEDURE , BYTE , PROC , VIRTUAL
END

WindowComponent INTERFACE
Kill                PROCEDURE
Reset               PROCEDURE (BYTE Force)
ResetRequired      PROCEDURE , BYTE          ! 1 if reset required
SetAlerts          PROCEDURE
TakeEvent           PROCEDURE , BYTE
Update              PROCEDURE                ! All but the window!
UpdateWindow       PROCEDURE
END
```

Notice the first interface has the `TakeRecord` method, the second `TakeEvent`. Now a given object can implement as many interfaces as it wishes, you can construct one object that implements both of these interfaces and your object can then function happily both as a record processor and as a component on a window.

Cloak And Dagger

Allow me to introduce you to what is probably the worlds most obscure "Hello World" program.



```

program
  map
  .

I1  INTERFACE
Talk  PROCEDURE(STRING S)
      END

I2  INTERFACE
Inform PROCEDURE( ),STRING
      END

Implementor CLASS, IMPLEMENTS(I1), IMPLEMENTS(I2)
      END

Matcher CLASS
Mix PROCEDURE(I1, I2)
      END

      CODE
      Matcher.Mix(Implementor.I1, Implementor.I2)

Implementor.I1.Talk PROCEDURE(STRING S)
      CODE
      MESSAGE(S)

Implementor.I2.Inform PROCEDURE
      CODE
      RETURN 'Hello World'

Matcher.Mix PROCEDURE(I1 Im1, I2 Im2)
      CODE
      Im1.Talk(Im2.Inform())

```

Going through in sequence you first meet two interface declarations. You can read them exactly as you would a class. The only difference is you will never see data. Each procedure declaration has an implicit `VIRTUAL` keyword and the `INTERFACE` is implicitly a `TYPE`. Now these interfaces are never created or actually *do* anything; they simply define a protocol (or interface!) that later objects will use.

Next comes an implementation class. This one implements one interface, although it could implement others and it could have methods of its own. The final `Matcher` class has a method that takes two interface parameters.

In the actual code the `Matcher` class receives two actual interfaces, both from `Implementor`. This is an important concept. Passing in an interface is equivalent to "handing over" a defined "bag" of methods that you are allowing the callee to call you back with. If `Implementor` had passed itself in then *all* `Implementor` methods would be available, but it doesn't, it only passes in those in the interface. If you look at the `Matcher.Mix` method you will find it simply calls a method of the first interface passing in a method result from the second.

So what have I gained for my obfuscation? Secrecy, or privacy if you prefer. The Matcher class has called two methods from the Implementor class *without knowing anything about the Implementor class!* Further the Implementor class doesn't know anything about the Matcher class.

I could (for example) implement a second class that only implemented the I2 interface to provide a French version of hello world (at least I could if I could remember the French for "World!").

```

program
  map
  .

I1  INTERFACE
Talk  PROCEDURE(STRING S)
      END

I2  INTERFACE
Inform PROCEDURE(),STRING
      END

Implementor CLASS, IMPLEMENTS(I1), IMPLEMENTS(I2)
      END

FrenchImplementor CLASS, IMPLEMENTS(I2)
      END

Matcher CLASS
Mix PROCEDURE(I1, I2)
      END

      CODE
          IF FrenchRequired
Matcher.Mix(Implementor.I1, FrenchImplementor.I2)
          ELSE
Matcher.Mix(Implementor.I1, Implementor.I2)
          END

Implementor.I1.Talk PROCEDURE(STRING S)
      CODE
          MESSAGE(S)

Implementor.I2.Inform PROCEDURE
      CODE
          RETURN 'Hello World'

FrenchImplementor.I2.Inform PROCEDURE
      CODE
          RETURN 'Bonjour '

Matcher.Mix PROCEDURE(I1 Im1, I2 Im2)
      CODE

```

```
Im1.Talk(Im2.Inform())
```

Now if you're following the plot you are probably thinking "but I can do this already; simply make `Mix` take an `Implementor` class as the parameter and then derive `FrenchImplementor` from `Implementor`." Which is true, you can, but why *should* you? I always like to consider the boundary between two classes as a fight (this works particularly well if you have two different people coding them!). What right does the person coding the `Matcher` class have to insist that the English and French coding teams have to base their work upon each other? Imagine the English and French versions are from different companies: this could cause chaos.

Now look at the `INTERFACE` solution. `I1` and `I2` are defined by the `Matcher` writer (typically `I1`, `I2` and the `Matcher` class would be defined in a header). That is all the `Matcher` defines and all it assumes. From then on anyone anywhere can implement as many or as few of those interfaces as it chooses and the `Matcher` will work with what it is passed.

The `WindowComponent` interface defined above provides a very practical example of this in `ABC`. Up to `C5B` the window manager had special knowledge burnt in about the `Browse` and `FileDrop` classes. Then it passed on certain events based upon knowledge of how the `Browises` and `FileDrops` worked. The code works fine but it is a real pain for us and for third parties. Suppose someone wants to add a component to a window that receives events from the window manager. First you have to decide if you are more like a `Browse` or a `FileDrop`, then you have to derive from one of those two (although you may be completely unrelated!) and pass yourself in to the window manager hoping you get the events you want.

Now from `C5.5` onwards the interface from the `WindowManager` is defined. The `FileDrop/Browse`-specific knowledge is removed from the `WindowManager` and anyone can plug into a `WindowManager` simply by implementing the `WindowComponent` interface. Remember that the interface only defines what methods can be called. It's up to the implementing class to decided what code gets executed inside those methods. The `WindowManager` doesn't have to know what it's talking to; it just has to know it can call certain methods, as defined in the interface.

If you want to blow your mind slightly more you will find the `browse` has been abstracted so that it doesn't rely on having a queue or driving a listbox! The queue and list control have both been given interfaces that can be implemented by anyone.

If you want to think of interfaces at a higher level they are really there to allow components to link together without getting too involved.

The Whole Hog

One little piece of the above syntax may be worrying some people: "Suppose the interface has ten methods; how do I show which ones to implement?" This is where we get a bit radical (and differ from Java). The answer is you do the lot!

The issue is really one of the implicit contract. An interface is, or should be, an explicit contract. You can tell what an object does by the interfaces it supports. The problem is that in real life the implementor of an interface will speak to a member of the target audience and say "Do you really need this function to support the `xyz` parameter? It is a real pain to implement and will threaten our Friday deadline." The user will typically say "No, I don't use that method at all".

There are now have two contracts in force, the explicit declaration of the interface and the implicit contract between the first implementor and user of the interface.

These implicit contracts are a great way to "make progress" but they slowly eat away at the stability of the object-oriented code base. To understand this you need to go back to what interfaces are for.

An interface defines a common boundary between two objects. If I am going to implement one side *without knowing about the other* I will feel at liberty to use each or all of those methods as I see fit. If I then only implement half of the methods because of implicit contracts I have negotiated, then when someone *else* plugs my implementation of the interface into a new usage it isn't going to work.

Java does have a solution to this although (in my opinion) it is horrible. Each interface grows a meta interface, where the meta interface effectively defines which methods of the actual interface are likely to work. In other words the meta interface defines the implicit contracts that an interface implementation is relying upon. Then when you are using an interface you end up with lots of code of the form :

```

if ( someinterface.canUseHighLevelMethod() )
    someinterface.HighLevelMethod();
else
{
    someinterface.lowlevelmethod1();
    someinterface.lowlevelmethod2();
}

```

Not only does this lead to code bloat at the user end but it leads to poor testing. You might have tested this code a hundred times but only against interfaces that have a HighLevelMethod that works; you are then plugged into to an interface that only has low-level methods and suddenly parts of your code are executing that have never been tested.

This is the sort of nightmare that has haunted ODBC users for years and threatens to do the same to JDBC users.

By enforcing that an interface implementation is complete, Clarion discourages implicit contracts in favour of explicit ones. The leads to smaller, more precisely defined interfaces.

Separation Cuts Both Ways (Or It Should!)

The Clarion implementation of interfaces aims to add one extra piece of hygiene to the equation that you don't get from C++ or Java: separation of the implementation from the interfaces.

Suppose I have two interfaces:

```

WC      INTERFACE
TakeEvent      PROCEDURE
            END
RP      INTERFACE
TakeEvent      PROCEDURE
            END

```

And a class that implements them

```

MyClass          CLASS , IMPLEMENTS ( WC ) , IMPLEMENTS ( RP )
                END

MyClass.WC.TakeEvent    PROCEDURE
                CODE
                DoSomething

MyClass.RP.TakeEvent    PROCEDURE
                CODE
                DoSomething

```

Then answer two questions for me:

1. If in the implementation of MyClass I want someone to take an event, which of these two methods should I call?
2. If I derive from MyClass which of these two methods do I override to affect all taken events? Consideration of this question should convince you that "who cares" is not a suitable answer to the first question!

These are questions you are faced with every time you implement two similar interfaces (maybe one modern, one legacy) in C++ or Java.

Clarion provides a sneaky solution by answering question #2 as "Neither, you cannot override an interface implementation (unless you re-implement the whole interface)." This pushes you towards a slightly different implementation of MyClass:

```

MyClass          CLASS , IMPLEMENTS ( WC ) , IMPLEMENTS ( RP )
TakeEvent        PROCEDURE , VIRTUAL
                END

MyClass.TakeEvent    PROCEDURE
                CODE
                DoSomething

MyClass.WC.TakeEvent    PROCEDURE
                CODE
                SELF.TakeEvent

MyClass.RP.TakeEvent    PROCEDURE
                CODE
                SELF.TakeEvent

```

The answer to question two now becomes that you always override down the class definition (not the interfaces). The answer to question one becomes that you always use the methods in the class definition, not the interface. Working in this manner MyClass can be seen to have *three* separate, clean, well defined interfaces, two external ones and an implementation interface. This also means that if the RP interface ever becomes redundant it can simply be ripped out without hurting the rest of the class.

Of course there is one potential downside in that you may have to do some busy work producing an implementation interface (Class definition) which can seem rather heavy

for a class that only implements one external interface.

By Their Fruits Shall Ye Know Them

I have said that interfaces can be used to allow an object to be used by multiple parents (or containers). I have said that interfaces can be defined by a container (or parent) to *define* objects (or children) it is capable of manipulating. I have said that Clarion interfaces can be used so that the implementation of an object is independent of the interfaces supported.

This leads to a situation where an object can be *defined* as the interfaces it supports whereby the implementation is a purely private matter for the programmer of the object. This enables an object to function in a completely alien environment simply by supporting the right interfaces. It also potentially allows an object that malfunctions in some aspects (a given interface is mis-implemented) to function correctly into other ways.

This finally brings us full circle. It is precisely this distinction that was used by Judas; he implemented a well-known interface (DiscipleGreeting), and the guards were primed to act upon the interface in a manner completely foreign to the original definition.

Of course I hope the metaphor doesn't carry too far. For Judas two-facedness and an internal undeclared error lead to destruction! My expectation instead is that interfaces will form a key part of a long term expansion and maintenance of the ABC system.

[David Bayliss](#) is a Systems Architect for The TopSpeed Development Center. He has worked upon TopSpeed's compiler and was the chief architect of the Application Builder Classes.

[Understanding OOP Interfaces](#)

(Jan 18,2000)

[WebBuilder Skeleton Basics II: Logos and Fonts](#)

(Jan 18,2000)

[xBase Y2K Driver Patches](#)

(Jan 18,2000)

[Advertise in Clarion Magazine!](#)

(Jan 18,2000)

Copyright © 1999-2000 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca.

published by

CoveComm Inc.

clarion magazine

Good help isn't that hard to find.**\$6.²⁵/month**[Main Page](#)[Log In](#)
[Subscribe](#)
[Renewals](#)[Frequently Asked
Questions](#)[Site Index](#)
[Article Index](#)
[Author Index](#)
[Links To
Other Sites](#)[Downloads](#)
[Open Source](#)
[Project](#)
[Issues in](#)
[PDF Format](#)
[Free Software](#)[Advertising](#)[Contact Us](#)**Feature Article**

January, 2000

Skeleton Basics II: Logos and Fonts

Steve Parker

Once I got over the thrill of seeing my first WebBuilder app running, I realized that there were several visual aspects of it with which I was not entirely satisfied. These included fonts, colors/backgrounds, listbox navigation buttons and listbox location. Later, I added logos to the list (not so much because I felt a pressing need for them as I realized that I had been prepared to provide this information).

Quite a list.

Some of these items could be controlled from within the .APP file. Fonts, for example, could be manipulated using embedded HTML. This is not necessarily convenient, to say the least (each and every control requires two embeds, each with exactly the same contents), but it *can* be done. In fact, all the items on my hit list can be effected from the skeletons and some can only be effected from there. But the bottom line is that the skeletons are where TopSpeed *expects* us to manipulate these items. Moreover, as it turns out, it really is easier in skeletons; it's a short learning curve and then just a few code snippets.

How do you do this? In this installment, I'll show you, starting with logos and fonts.

Logos

Logos are the easiest skeleton modification to implement (and the easiest to see).

What I need to do is to place a logo at the top and/or bottom of every page (or, obviously, both). Of course, I can embed the image at the top and/or bottom of every window in the .APP (see "Images in Internet Connect," available at www.par2.com/ic/images.htm). But this doesn't seem very much like rapid application development and enhancement, does it?

What I want is to "set and forget," do it once and be done with it. In fact, this is also the concept underlying skeletons. The notion is that modifications should be able to affect

[Understanding OOP
Interfaces](#)
(Jan 18,2000)[WebBuilder Skeleton
Basics II: Logos and Fonts](#)
(Jan 18,2000)[xBase Y2K Driver Patches](#)
(Jan 18,2000)[Advertise in Clarion
Magazine!](#)
(Jan 18,2000)[Skeleton Basics
Part 1](#)[Skeleton Basics
Part 2](#)[Skeleton Basics
Part 3](#)[Skeleton Basics
Part 4](#)**TopSpeed****Clarion 5**
by TopSpeed



not only a single page, nor even all the pages generated by a single app, but all of the pages in every app on a site (or, to be precise, all apps using a common skeleton set regardless of their location) and to do so from a single file/file set.

As I indicated in my previous article ([Skeleton Basics: Part 1](#)) I first thought that to show a logo, the skeleton to be modified had to be WinCore.HTM. Since WinCore is the skeleton which controls the basic visual characteristics of every page, the inference seems obvious.

In fact, logo placement *can* be accomplished in WinCore.HTM (of course). But this skeleton is already a pretty busy place. If only for the sake of readability, it would be nice if there were there somewhere else I could do this. (A bit of advance word: I am advised that Window.HTM, WinCore.HTM, ColorA.HTM and ColorB.HTM will be combined in a future release. Everything else said here is still applicable, except, of course for the skeleton name.)

As it turns out, a bit of playing about demonstrated that modifying Window.HTM will also work quite nicely:



```
<!-- Window.htm -- Start -->
<html>
<head>
<meta name="ts-control" content="window,application">
</head>
<!-- images here appear at top of each page -->
<TSINCLUDE name="colora.htm">
<TSINCLUDE name="wincore.htm">
<TSINCLUDE name="colorb.htm">
<!-- images here appear at bottom of each page -->
</html>
<!-- Window.htm -- End -->
```

This particular modification works perfectly for images (and any other HTML, such as copyright notices and the like) at the bottom of the page, a little less so for images at the top.

Images placed at the top of the page will cause a JavaScript error when the user presses the close button supplied by the skeletons to terminate the app. Of course, since many users don't bother ever closing apps (including many of you when visiting my [knowledge base](#), for example), this may not be an issue.

If you are concerned that your code execute, under all circumstances, just as cleanly as possible, however, this is an issue. You have two choices.

First, you can remove this button and its functionality. I think that for many apps, this is entirely appropriate. Problem resolved; with the close button is removed, a top logo in Window.HTM will function perfectly.

An interesting thought is that you could use a small logo in place of the standard x.gif with or without the close functionality. On reflection, however, I suspect that retaining close functionality might cause a customer to look askance at their logo being used to terminate their app. Okay, substitute a logo for x.gif but remove the functionality.

The relevant section from WinCore.HTM is:

```

<table finalcolor="Border" border="0"
  cellpadding="4" cellspacing="2" width="100%">
  <tr finalcolor="Header"> Ç Title area, begin
    <td width=100%>
      <b>
        <TSSCRIPT value=Title> Ç Title substitution, begin
          Page Title
        </TSSCRIPT> Ç Title substitution, end
      </b>
    </td> Ç Title area, end
    <td width=0%> Ç Close button area, begin
      <TSSCRIPT tag=a attr=href replace="NAME"
        value="Name">
        <a href="javascript:icSubmit
          ('NAME$EventCloseWindow');"> Ç script,
          begin
          <img alt="Close" WIDTH="18" HEIGHT="15"
            SRC="/x.gif" BORDER=0> Ç image
          </A>
        </TSSCRIPT> Ç script, end
      </td>%> Ç Close button area, end
    </tr>
  </table>

```

Taking all of this a step or two further, if you don't want the title bar (which is equivalent to `window{Prop:Text}` and I expect many of you don't want it on your web apps), simply remove the entire row, all of the code between `<tr>` and `</tr>`, inclusive. Alternately, you can place your logo where the "Page Title" is currently located, removing only the two TSSCRIPT tags and placing the image reference where "Page Title" appears.

Second, noting that `Window.HTM` does not have a `<BODY>` tag (which is the source of the Javascript problem), you can place the image reference immediately after the `<BODY>` tag in `WinCore.HTM`:

```

</head>
<body finalcolor="Page" bgcolor="white" ...
<!-- images here appear at
  the top of each page -->
<TSSCRIPT value="EmbedBeforeWindow" type=html>
</TSSCRIPT>

```

If the existing close functionality is retained, use this method for placing a top logo.

Now, that wasn't hard, was it?

So much for logos.

Fonts

The default font on my browsers is Times Roman. This is fine for memos and other documents but web pages don't look very good in this font.

Almost nothing is easier to do than changing a font in a visual HTML editor.

Simply block the text to be changed and select the font from the drop down list. You're done. When creating a new document and setting the default font, skip the "block" step. Done at step one, as they say.

It is not quite so easy with skeletons and cannot be. This is because each skeleton controls the production of HTML for a single type of control. Therefore, each type of control that could produce string output needs to be modified.

Employing the techniques outlined in the previous article, one quickly discovers that the skeletons that need to be modified include:

- Check.HTM – the text for a checkbox
- DisplayText.HTM – the actual text for both string and prompt controls
- Sheet.One.HTM – selected/unselected text on tabs
- Tab.All.HTM – tab text
- Table.HTM – text in list boxes

SString.HTM also appears to contain text but I am unsure when it is used. And there may be other skeletons that generate string data that I've missed. However, employing the technique I outlined last time and searching for the comments left by the skeletons, it will be easy enough to discover any skeletons that require further attention.

The only "trick" is to place the HTML outside the <TSSCRIPT> tags. For example:

```

<!-- DisplayText.htm -- Start -->
<TSSCRIPT tag=font attr=color value="DisabledText"
  when="Disabled">
<FONT face="verdana,arial,sans-serif">
<TSSCRIPT value=DisplayText>

```

This is the text:

```

</TSSCRIPT>
</FONT>
</TSSCRIPT>
<!-- DisplayText.htm -- End -->

```

(This skeleton is available for [download](#).)

TSSCRIPT is used to replace code. Therefore the TSSCRIPT tags indicate an area of HTML substitution. If you place any new HTML inside TSSCRIPT tags, that HTML will disappear (and the nominated - that is, included - text along with it, as I discovered the hard way).

Also, since you cannot be sure that any given machine will have the font you most wish to use, it is wise to supply alternatives, as shown. The font tag shows two specific sans-serif fonts, as well as the sans-serif font family as a last resort.

Style Sheets

At face value, the approach I've described appears to be a remarkably inefficient way to modify the font for a skeleton set. And, indeed, if you want to use a single font for an entire app or site, it is. (Besides, it just doesn't sound very "Clarion" to have to modify and maintain so many individual files.)

The answer is style sheets, which let you specify a font (and other attributes) for any HTML tag. The exception is if, say, you want radio buttons in one font and checkboxes in another, or prompts in a different font from everything else or listboxes visually differentiated. In those cases a Style Sheet won't help. If you

need or want one text-producing skeleton to use a different font, you must modify the skeleton that creates that control. In this case, there really is no option.

NOTE: If you are not familiar with Style Sheets, you may find the primer at <http://builder.cnet.com/Authoring/CSS> useful. This URL is case sensitive.

If you do want a single font, Style Sheets (or, more properly, Cascading Style Sheets, a.k.a. CSS) should make the assignment of a font to the apps on a site much more efficient than modifying individual skeletons.

And, indeed, this is the case.

Except not all browsers support Style Sheets or don't support style sheets fully.

If you do not intend to fully support pre-4.0 level Netscape or Microsoft browsers or do not have to fully support them, this is not an issue. That is, if you can control the client browser or are willing to allow users of older or non-compliant browsers to judge your work by the browser's default font, there is no reason not to use CSS and to do so exclusively. But, if you do want to control older browsers, you will need to modify the individual skeletons, as discussed above. (And be warned that Netscape 4.x's support for style sheets is a bit sloppy, so if you have to support Netscape (and most of us do) you will face some limitations.)

Except there is no obvious selector (tag) for the Style Sheet. Style Sheets must be attached to a tag, which becomes its selector. For instance, each paragraph on this page begins has the P tag: it begins with <p> and ends with </p>. (And, this is really very clever, really quite Clarion-like: any time a certain kind of HTML is used, CSS triggers the use of a specified set of attributes and reminds me of field priming on insert.)

But normal page literals (strings) in WebBuilder-generated HTML do not have explicit tags.

My first thought was to try attaching a Style Sheet to the <BODY> tag, since all text in an HTML page must be between the <BODY> and </BODY> tags. Something in my reading, however, indicated that this might not work. Later, other reading indicated that it might. In fact, I still do not know whether a Style Sheet will work on the <BODY> tag or not. And, as it turns out, it is not important whether it does.

What is important is that understanding the way in which WebBuilder creates HTML shows that there is a ready-built place to attach Style Sheets and to do so very close the final text.

Specifically, most Clarion-generated HTML is inside a <FORM> but all of it is inside a <TABLE>.

<TABLE> is a tag and, therefore, a candidate selector. But wait, there's more.

Inside an HTML <TABLE>, there are rows (<TR> tags) and data is actually presented within Table Data (<TD>) tags (the area bounded by Table Data tags is also referred to as a "cell"). Clarion generated HTML liberally employs all of these tags.

Therefore, a Style Sheet, similar to:

```
<style type="text/css"><!--
td{font-family: arial,verdana,sans-serif}
-->
</style>
```

can be used to set the font for the <td> tag. Moreover, this is the tag where

strings are actually output (indeed, if you use a visual HTML editor to change the font of a table, it will be applied at the cell level, exactly as this CSS will). How very convenient. (A modified WinCore.HTM skeleton containing this Style Sheet is available for [download](#).)

What about that one control you need in a different font? If I understand CSS correctly, if the skeleton for that control type specifies a font, it will override the Style Sheet (though I haven't tested this).

Summary

Not rocket science, this. But it is a new way of doing things for most of us. It does require some familiarity with HTML but not necessarily expertise.

Next time, I'll continue investigating skeleton customization with colors and backgrounds.

[Download the zip](#)

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

[Understanding OOP Interfaces](#)

(Jan 18,2000)

[WebBuilder Skeleton Basics II: Logos and Fonts](#)

(Jan 18,2000)

[xBase Y2K Driver Patches](#)

(Jan 18,2000)

[Advertise in Clarion Magazine!](#)

(Jan 18,2000)

Copyright © 1999-2000 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca.

Clarion MAGAZINE

Clarion
Development
Resourcespublished by
CoveComm Inc.clarion magazine
Good help isn't that hard to find.**\$6.²⁵/month**[Main Page](#)[Log In](#)
[Subscribe](#)
[Renewals](#)[Frequently Asked
Questions](#)[Site Index](#)
[Article Index](#)
[Author Index](#)
[Links To
Other Sites](#)[Downloads](#)
[Open Source
Project](#)
[Issues in
PDF Format](#)
[Free Software](#)[Advertising](#)[Contact Us](#)

Downloads

December, 1999

Free Software

The following non-open-source free software is available from Clarion Magazine:

Gordon Smith's [Class Browser](#)

This indispensable utility lets you browse Clarion class hierarchies.

[setupcb.exe](#) - 560kb

Gordon Smith's Batch Compiler

This batch compiler automates compiles and is typically used for multi-dll applications.

[setup16.exe](#) - 771kb

Check the [Open Source Project pages](#) for other free software.

xBase Y2K Patches For CFD 3102 And CW 2003

Earlier versions of Clarion have problems reading xBase files when the file header has been updated incorrectly by another program for dates greater than 1999. These unsupported patches are intended to allow Clarion to work with such xBase files. **Use at your own risk! These patches come from a non-TopSpeed source and are not guaranteed in any way at all!**

[Y2K xBase Patch For CFD 3102](#)

[Y2K xBase Patch For CW 2003](#)

[About Our New Look](#)
(Dec 7, 1999)[Special Report: ConVic '99
Clarion Conference](#)
(Dec 7, 1999)[Template Writing Made
Easy](#)
(Dec 7, 1999)[The SQL Answer Cowboy](#)
(Dec 7, 1999)[The Clarion Advisor -
Listbox Styles](#)
(Dec 7, 1999)[The Cranky Programmer:
Nits And Bits](#)
(Dec 7, 1999)[ABC Design Series: View
Manager Part 2](#)
(Dec 7, 1999)[New ClarionMag Link
Buttons](#)
(Dec 7, 1999)[ClarionMag Newsgroup
Reminder](#)
(Dec 7, 1999)[Free Utilities: Class
Browser And Compile
Manager](#)
(Dec 7, 1999)[Advertising Packages](#)
(Dec 7, 1999)

Copyright © 1999-2000 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca.

Clarion MAGAZINE

Clarion
Development
Resourcespublished by
CoveComm Inc.

clarion magazine

Good help isn't that hard to find.**\$6.²⁵/month**[Main Page](#)[Log In](#)
[Subscribe](#)
[Renewals](#)[Frequently Asked Questions](#)[Site Index](#)
[Article Index](#)
[Author Index](#)
[Links To Other Sites](#)[Downloads](#)
[Open Source Project](#)
[Issues in PDF Format](#)
[Free Software](#)[Advertising](#)[Contact Us](#)

Clarion Magazine Information

December, 1999

Advertising In Clarion Magazine

Why advertise in Clarion Magazine? With over 4000 visits per month (that's 50,000 per year!), Clarion magazine is your number one Internet opportunity to present your product to Clarion developers.

Here's what Kathryn and Phil Carroll of [Paragon D&D](#) have to say about advertising in Clarion Magazine:

At [Paragon D&D](#), we think our Clarion Magazine ad definitely brings us new business. Our web site logs indicate that several recent UltraTree customers were referred to our site directly from Clarion Magazine. In fact, we receive more web referrals from Clarion Magazine than from any other source, including our accessory listing on the TopSpeed web site. The ad space is generous, and the exposure is second to none!

Advertising in Clarion Magazine is a cost-effective way to reach your target market. It's the best Clarion web exposure you can buy, and it's easy on your budget.

If you'd like to place an ad, or you have questions not answered below, contact ads@clarionmag.com.

Policies

Advertising on Clarion Magazine is subject to various conditions and costs. Please review all of the information on this page before submitting an ad.

Contents

[Rates](#)[Clarion Magazine Main Page](#)
(Feb 12, 1999)[The Clarion Magazine FAQ](#)
(Feb 24, 1999)[The Site Index](#)
(Feb 12, 1999)[Subscribe to Clarion Magazine!](#)
(Feb 8, 1999)[The Subscription Agreement](#)
(Feb 5, 1999)[Refund Policy](#)
(Feb 8, 1999)[Subscriber Services](#)
(Feb 8, 1999)[Information For Authors](#)
(Feb 2, 1999)[How To Contact Clarion Magazine](#)
(Feb 2, 1999)[The Unofficial Clarion OOP Page](#)
(Feb 6, 1999)[Lend Us Your Links](#)
(Feb 7, 1999)[Getting Access To Clarion Magazine](#)
(Feb 7, 1999)[Clarion Magazine's Automated Mailing Lists](#)
(Jan 31, 1999)[Clarion Magazine Is Best Read With Verdana](#)
(Feb 7, 1999)[Clarion Links By Category](#)
(May 17, 1999)[Free Newsgroups For Clarion User Groups](#)
(May 10, 1999)[Free Utilities: Class](#)

[Image Type](#)
[Content](#)

Rates

Currently there are five kinds of ad spaces available on Clarion Magazine. They are main page sidebar, article sidebar, news pages, main page banner, and index pages. All rates are effective January 13, 2000. Sidebar ads are charged by height - if you choose a size other than those suggested your costs will vary accordingly.

Main Page Sidebar Ads

Main page sidebar ads appear on the right side of the main menu, in random order (which changes each time the page is accessed). Ads may be no more than 115 pixels wide and may be any reasonable height. There are a limited number of spaces available. The rate is **\$75 per month** for a typical ad size of **115 x 175**.

Article Sidebar Ads

Article sidebar ads may be no more than 115 pixels wide and may be any reasonable height. These ads appear on the left margin of feature articles and selected other pages. For a typical ad size of **115 x 175 pixels** the rate is **\$50 per month**. These ads remain permanently with the articles for which they are purchased. The order ads appear on the page is at the discretion of Clarion Magazine, and is normally on a first come, first placed basis. Since article ads are only seen by subscribers, you know you're targeting people who are willing to buy Clarion-related products.

News Page Ads

News page ads may be no more than 115 pixels wide and may be any reasonable height. These ads appear on the left margin of news pages. For a typical ad size of **115 x 175 pixels** the rate is **\$50 per month**. These ads remain permanently with the news pages for which they are purchased. The order ads appear on the page is at the discretion of Clarion Magazine, and is normally on a first come, first placed basis. News pages are publicly accessible and are viewed by subscribers and non-subscribers.

Main Page Banner Ads

[Main page](#) banner ads must be 468 x 60 pixels. This ad is available for **\$75 per week or \$250 per month**. Banner ad placements are not yet available on other pages except by special arrangement. For more information contact ads@clarionmag.com.

Index Page Ads

Index page ads appear on the [site index](#), [author index](#), and [article index](#) pages. For a typical size of **115 x 175 pixels**, the rate is **\$50 per month**. These ads do not remain permanently with the page.

Image Type

[Browser And Compile Manager](#)
(Dec 7, 1999)

[Alphabetical Author Index](#)
(Jan 4, 2000)

[Alphabetical Article Index](#)
(Jan 4, 2000)

[Clarion's Publication Schedule](#)
(Feb 5, 1999)

[Advertising Policy](#)
(Feb 7, 1999)

[Open Source Code Available Now](#)
(Mar 8, 1999)

[The Masthead](#)
(Feb 15, 1999)

Images can be in GIF or JPEG format, and the palette requirement has been dropped on the assumption that just about everyone uses more than 256 colors.

At present we do NOT accept animated ads.

Advertising Content

Ads are accepted for all products and events of interest to Clarion developers, assuming the ads are of suitable quality and acceptable content, and do not conflict with the purpose and mission of Clarion Magazine.

For further information send an email to ads@clarionmag.com.

Copyright © 1999-2000 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca.

Clarion MAGAZINE

Clarion
Development
Resourcespublished by
CoveComm Inc.

clarion magazine

Good help isn't that hard to find.**\$6.²⁵/month**[Main Page](#)[Log In](#)
[Subscribe](#)
[Renewals](#)[Frequently Asked
Questions](#)[Site Index](#)
[Article Index](#)
[Author Index](#)
[Links To
Other Sites](#)[Downloads](#)
[Open Source
Project](#)
[Issues in
PDF Format](#)
[Free Software](#)[Advertising](#)[Contact Us](#)

Feature Article

January, 2000

Understanding Clarion Code Part 3

by David Harms

At the close of the previous article in this series I discussed a small utility program (written by Carl Barnes) which is used to delete old temporary files from the Windows directory. That utility doesn't present any user information except a message indicating it has finished processing. As such, although useful, it isn't really representative of the kinds of programs most of us create using the Application Generator.

In this article I'll take the temp file utility a step further by adding a window to display the files which are to be deleted, and to give the user Delete and Cancel options.

Creating A Procedure

To begin with I'll move the code into its own procedure and source module. This isn't strictly necessary in a very small utility like this which only has one function. However this is essentially what the AppGen does and it's essential for larger programs if you want to keep compile times to a minimum.

Moving code into its own procedure is quite easy. First, you add the procedure name you want to use to the map statement:

```
MAP
    Test ( )
END
```

Then, after the code statement, add the procedure call:

```
CODE
Test ( )
```

Finally, enter the procedure declaration and a code statement before the actual executable code:

```
Test          procedure
```

[Understanding Clarion
Code
Part 1](#)[Understanding Clarion Code
Part 2](#)

...

[The Novice's Corner:
Clarion Code 3](#)
(Jan 25,2000)[Skeleton Basics III: Colors
and Backgrounds](#)
(Jan 25,2000)[January 2000 News](#)
(Jan 25,2000)[The Cranky Programmer](#)
(Jan 25,2000)

CODE

The source code now looks like Figure 1.

Figure 1. The utility program updated to use a procedure.

```

PROGRAM

    MAP
        Test ( )
    END

DirQ      QUEUE(ff_:Queue) ,PRE(AnythingAtAll)
END

Count     LONG
Idx       LONG
TempDir   STRING(255)

CODE
Test ( )

Test      procedure
code
TempDir = 'C:\windows\Temp'
!TempDir = 'C:\Temp'
Count = 0
DIRECTORY(DirQ,CLIP(TempDir)&' *.*',ff_:Normal)
LOOP Idx = 1 TO RECORDS(DirQ)
    GET(DirQ,Idx)
    IF DirQ.Date < TODAY() - 4
        REMOVE(CLIP(TempDir)&'\'&DirQ.Name)
        Count += 1
    END
END
MESSAGE('Done! ' & Count & ' file(s) removed')

```

So what's really happened here? Not a whole lot, yet. The first CODE statement in the main module, which is the one with the PROGRAM statement at the top of the file, is where program execution starts. That CODE statement used to lead straight into the utility code; now it leads to a Test () procedure call. All Clarion AppGen programs do exactly the same thing, except they typically have a few lines of initialization code, followed by a call to Main () (or whatever you choose to call your main menu procedure), followed by a bit of code to clean things up. Figure 2 shows some typical AppGen program startup code.

Figure 2. Typical code to start up an ABC AppGen application.



```
CODE
GlobalErrors.Init
INIMgr.Init('cmsubs.INI')
DctInit
Main
INIMgr.Update
INIMgr.Kill
DctKill
GlobalErrors.Kill
```

In Figure 2 the call to `Main` doesn't use an empty parameter list, while in the utility example `Test()` does. You can write the call either way; normally it doesn't make any difference. I like using `()` at the end of a procedure call at all times for the sake of consistency.

Back to the example. There's one thing about Figure 1 that I really don't like. All the data is global, because it's declared before the program's `CODE` statement. That means that if I create additional procedures, they will all share that globally declared data.

Now, everyone, repeat after me: "Global Data Is Bad! Bad Global Data! Very Bad!" Here's why. In `Test()`, for example, the variable `Idx` is used as a counter. If I called another procedure (there isn't one, but I might create one later), that procedure could conceivably also use the `Idx` counter. If I called that procedure from `Test()` it might change the value of `Idx`, and if I called it in the middle of a loop the results could be disastrous. I might end up in an infinite loop, trying to delete non-existent files.

It's true that almost every Clarion application needs some global data. Files are typically declared globally, although that's also caused a lot of grief over the years. For more read the article by David Bayliss on the `FileManager`, called "[Propitious Memory Corruption](#)." You may also have come across the `GlobalRequest` and `GlobalResponse` variables, which are used to communicate information between procedures. In both these cases the AppGen generates, or in the case of ABC makes us of, a certain amount of code to manage the problems of sharing data globally. So while some kinds of global data are a necessary evil, you should always think carefully about any data you declare globally. And be very careful how you use that data.

Following the practice described in [Part 1](#) of this series move the data into the `Test()` procedure, and move `Test()` to its own module. The code will now be in two files, as shown in Figures 3 and 4.

Figure 3. The DELTEMP.CLW module.

```

PROGRAM

    MAP
        MODULE( 'DELTEMP1.CLW' )
            Test()
        END
    END

    CODE
    Test()

```

Figure 4. The DELTEMP1.CLW module.

```

    MEMBER( 'DELTEMP.CLW' )

Test      PROCEDURE

DirQ      QUEUE(ff_:Queue),PRE(AnythingAtAll)
          END

Count     LONG
Idx       LONG
TempDir   STRING(255)

    CODE
    !TempDir = 'C:\windows\Temp'
    TempDir = 'C:\Temp'
    Count = 0
    DIRECTORY(DirQ,CLIP(TempDir)&' *.*',ff_:Normal)
    LOOP Idx = 1 TO RECORDS(DirQ)
        GET(DirQ,Idx)
        IF DirQ.Date < TODAY() - 4
            REMOVE(CLIP(TempDir)&'\'&DirQ.Name)
            Count += 1
        END
    END
    END
    CLOSE(Window)

```

Remember that you'll have to add DELTEMP1.CLW to the project so the compiler knows where to find it. Also DELTEMP.CLW has to have the MEMBER('DELTEMP.CLW') statement at the top so the compiler knows where to find the procedure declaration (and any global data, should you be so bold).

Creating A Window

Let's say you want to display which temporary files are going to be deleted, rather than just killing them off blindly. Fortunately, creating a window to display data is easy. Just

position your cursor in the procedure's data area (that is, before the CODE statement) and press Ctrl-F, or choose Edit|Format Structure from the menu. You'll be presented with a list of Window and Report structures to choose from (these are defined in DEFAULTS.CLW in your LIBSRC\ directory).

In this case I want a window, and as it happens the kind I want is first on the list. The other choices are: Window with OK and Cancel buttons (I'll be adding my own buttons, so that's not needed); System Modal Window, which in 16 bit programs means you can't switch to other programs while the window is open (not usually a good idea); MDI Child Window, which can't be opened without an MDI frame; MDI Parent Frame, good for main menus but you can't put any controls on the window itself, so not useful here; and finally, a System Resizable Window.

When you just want to open a simple, single window for an application, don't use MDI. Choose a non-MDI window (like Window). You can always change the window attributes later if you want.

If you've chosen the Window default you should see that window in the window formatter. Go to the window properties (right click on the window and choose Properties) and set the frame type to Double. You can also set the caption to something like "Delete Temporary Files." Exit, saving your changes. You'll see a window structure something like the following in the source:

```
Window WINDOW('Delete Temporary Files'),AT(, ,238,162), |
        GRAY,DOUBLE
```

END

To display the window, at a minimum you only need the following code:

```
OPEN(Window)
ACCEPT
END
CLOSE(Window)
```

After the window is opened you need to have an ACCEPT loop. The ACCEPT statement is a fairly large black box that handles the sorts of things that you and I generally don't need to mess with, such as drawing/redrawing the window, responding to CloseWindow events and so forth. ACCEPT is both a blessing and a curse. It makes life a lot easier most of the time, but it means that if you want to intercept some of those low-level messages you have to go through the extra step of subclassing (there's an example of this in [Pierre Tremblay's article](#) on sliders).

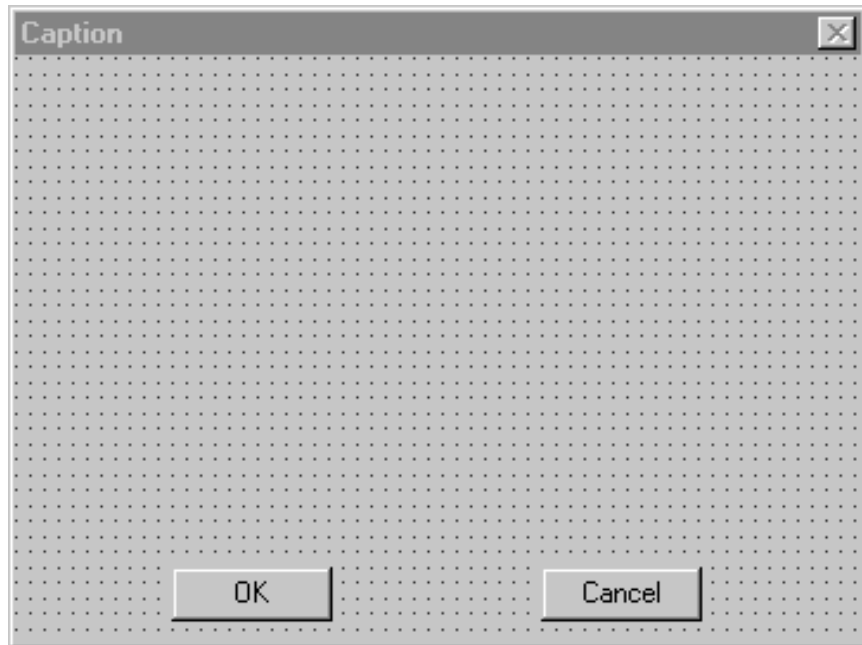
This minimal window structure and ACCEPT loop isn't all that useful. Ideally this window should display the files to delete and offer the user the choice to proceed.

Start with the buttons. Place the cursor anywhere in the window structure and press Ctrl-F. In the window formatter, add two buttons. For each button right-click on the button to bring up the property dialog, or use the property toolbox. For one button set its Text to Cancel and its Use field to ?Cancel; for the other use OK and ?OK.

Broadly, there are two types of windows available: MDI and SDI, which stand for Multiple Document Interface and Single Document Interface. SDI is more commonly called non-MDI. MDI windows are of two types, parent and child. Parent MDI windows are also called application frames, and are what you normally use as a main menu in a Clarion application. MDI child windows have to be started after the frame, and always stay within the bounds of the frame. Non-MDI windows will display outside the bounds of the application frame.

Also, for the sake of appearance, right-click on the window, choose Font, and set the window's font to something like MS Sans Serif Regular 8. Your window should look like Figure 5.

Figure 5. The window with OK and Cancel buttons.



Save your changes. Next you need to place some code inside the ACCEPT loop to respond to the user clicking on those buttons. The ACCEPT loop can do a lot on its own, but it can't anticipate what kind of controls you're going to put on the screen. Figure 6 shows the ACCEPT code.

Figure 6. Using the ACCEPT loop to handle user actions.

```

ACCEPT
CASE ACCEPTED( )
OF ?OK
  LOOP Idx = 1 TO RECORDS(DirQ)
    GET(DirQ,Idx)
    IF DirQ.Date < TODAY() - 4
      REMOVE(CLIP(TempDir)&'\'&DirQ.Name)
      Count += 1
    END
  END
  MESSAGE('Done! ' & Count & ' file(s) removed')
  POST(Event:CloseWindow)
OF ?Cancel
  POST(Event:CloseWindow)
END
END

```

There are several new features in this block of code. One is the use of a CASE statement. CASE is a lot like an IF...ELSIF...ELSIF...END structure but it has the advantage that the condition to be tested (in this case the result of the ACCEPTED function) only has to be evaluated once. CASE stores the result and evaluates it against

each OF test, and if it finds a match, executes the code following the OF.

The ACCEPTED function is part of Clarion and returns a number indicating which control was accepted. But where does this number come from?

Look back at the window structure and you'll remember that each button has both a text attribute and a USE attribute. For buttons (and most other kinds of controls) the use attribute is prefixed with a ? in the window structure. For entry fields and other controls which store a value in a variable, however, the use attribute will actually be the variable, without the ?, which the entry field is to be updated.

Each control on the window actually has an internally-assigned number, with the first control (by default) having the number 1 (menu items and toolbar controls use negative numbers starting with -1). Use attributes, more commonly called use variables, are a way of referencing the field number without having to know what the actual number is. You don't normally want to use the numbers directly because shuffling the order of the controls, or adding a new control at the top, would change those numbers and break your code.

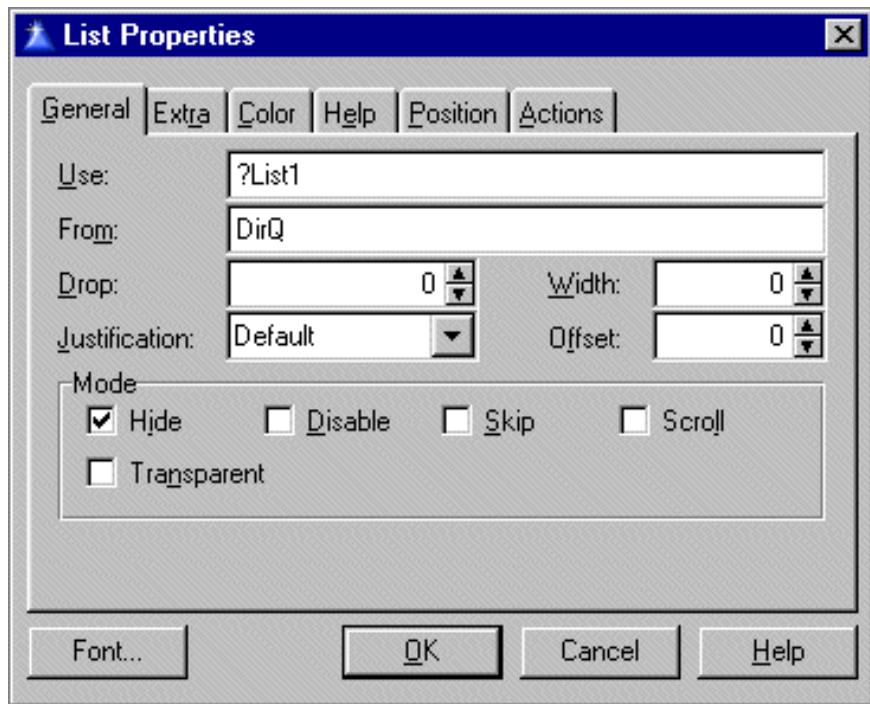
For controls which don't use a variable to store a value the name of the use variable doesn't have to be the same as the text; it can be anything you want. But it's helpful to make the use variable something easily recognized.

In the example code, the CASE structure matches code up with an ACCEPTED event on a particular button. One button triggers the delete, but in both cases the window should be closed, and this is normally done by using the POST function to send the CloseWindow event to the window. The ACCEPT loop then terminates, the window is closed, and the procedure (and ultimately the program, in this case) exits.

Adding A List Box

Open the window again in the window formatter. Select Control|List Box from the menu or choose the list box icon on the controls toolbox and place a list box on the window. Resize it appropriately. Right click on the list box and choose Properties. As shown in Figure 7, set the From field to DirQ, and on the Extra tab enable the Horizontal and Vertical Scroll Bars.

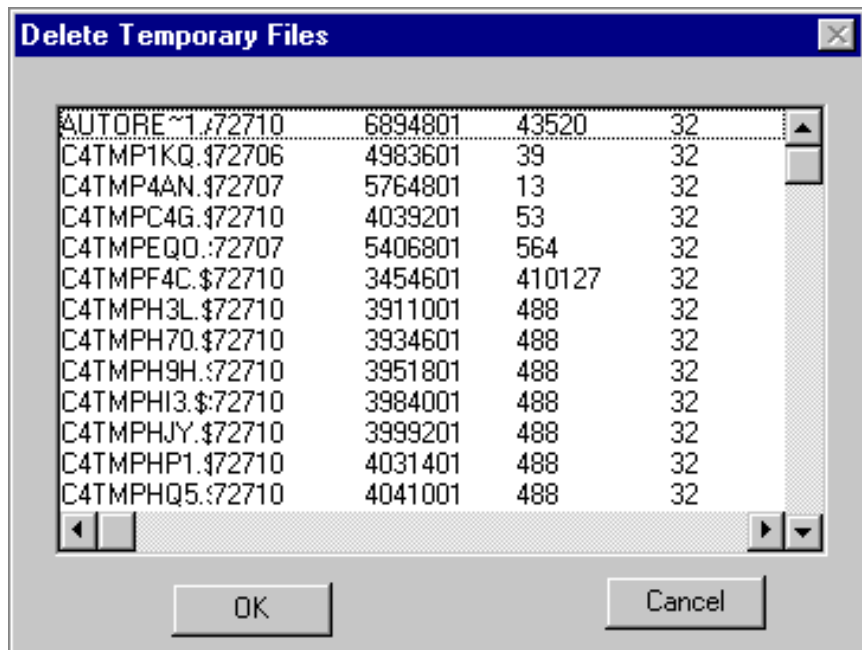
Figure 7. List box settings for displaying DirQ.



Save and compile. Did you get an error that said "Unknown Identifier: DirQ?" If you did, it's because `DirQ` is declared after the window. It needs to be moved before the window. The Clarion compiler does allow forward referencing in some situations, but this isn't one of them.

Fix the problem, and compile and run the program. If you've correctly specified your temp directory, and it contains some files, then you should see a list box like Figure 8.

Figure 8. A crude list box displaying files in the temp directory.



This list box doesn't look like much. But then again, consider how little code you needed to add. The list box control is declared in the window structure like this:

```
LIST, AT(12, 12, 216, 116), USE(?List1), HVSCROLL, FROM(DirQ)
```

All that's connecting `DirQ` to the list box is the `FROM(DirQ)` attribute. The list box

looks at the queue structure and displays each field in the queue as a string. Not bad for one line of code.

Summary

This utility still has a ways to go. The list box should be formatted, and as you've probably noticed it's displaying all of the temp files, not just the ones that are candidates for deletion. And then it might be nice to be able to resize the list box, or to mark which files ought to be deleted. That's the problem with writing software; feature lists never stop growing.

I'll look at some of these issues next month. In the meantime, if you have some modifications to this utility you'd like to share, email your code to me at dharms@clarionmag.com.

[Download the source code](#)

[David Harms](#) is an independent software developer and the co-author with Ross Santos of *Developing Clarion for Windows Applications*, published by SAMS (1995). He is also the editor and publisher of Clarion Magazine.

[The Novice's Corner: Clarion Code 3](#)

(Jan 25,2000)

[Skeleton Basics III: Colors and Backgrounds](#)

(Jan 25,2000)

[January 2000 News](#)

(Jan 25,2000)

[The Cranky Programmer](#)

(Jan 25,2000)

Copyright © 1999-2000 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca.

published by

CoveComm Inc.

clarion magazine

Good help isn't that hard to find.

\$6.²⁵/month

[Main Page](#)

[Log In](#)
[Subscribe](#)
[Renewals](#)

[Frequently Asked Questions](#)

[Site Index](#)
[Article Index](#)
[Author Index](#)
[Links To](#)
[Other Sites](#)

[Downloads](#)
[Open Source](#)
[Project](#)
[Issues in](#)
[PDF Format](#)
[Free Software](#)

[Advertising](#)

[Contact Us](#)

Feature Article

January, 2000

Skeleton Basics III: Colors and Backgrounds

Steve Parker

Editor's note: This is, by all accounts, Steve Parker's 75th birthday...ah, wait, it's his 75th *article* for Clarion publications, including Clarion Tech Journal, Clarion for Windows Journal, Clarion Online, and of course Clarion Magazine. At his present rate of production he should hit 100 by spring. Way to go, Steve!

When you think of customizing a web page, colors and background images are probably among the first things that come to mind. So it's time to look at customizing colors and backgrounds in the skeletons.

Colors

Two skeletons are used to implement colors in a WebBuilder application, ColorA.HTM and ColorB.HTM. (In future releases, these two files will be merged into Window.HTM, along with WinCore.HTM.)

ColorB "only" supplies the terminating tags for color assignment HTML and, therefore, should be neither touched nor deleted. Modifying ColorB.HTM would most likely end up creating HTML with mismatched tag pairs. Mismatched tag pairs are quite likely to cause runtime failures in browsers, particularly Netscape, which shares the "defect" of expecting properly formatted code (though this is one bug that Internet Explorer does not seem to have in common with the more senior browser, it too will fail to display the page(s) in this circumstance).

ColorA.HTM, by elimination then, contains the operating implementation of colors. ColorA is a somewhat more complicated file than the other skeletons. While it is not essential to master the details of its structure, a quick look will be most useful.

Other skeletons are composed of HTML, HTML plus INCLUDEs or HTML plus runtime substitutions (TSSCRIPT). The most complex feature of most other skeletons is the use of runtime switches, branching code, to differentially produce HTML (e.g., an entry or,

[The Novice's Corner: Clarion Code 3](#)
(Jan 25,2000)

[Skeleton Basics III: Colors and Backgrounds](#)
(Jan 25,2000)

[January 2000 News](#)
(Jan 25,2000)

[The Cranky Programmer](#)
(Jan 25,2000)

[Skeleton Basics Part 1](#)

[Skeleton Basics Part 2](#)

[Skeleton Basics Part 3](#)

[Skeleton Basics Part 4](#)

TopSpeed

Clarion 5
by TopSpeed

FREE Microsoft Internet Explorer

etc 2000
EVENT SPONSOR



if the control has the REQ attribute, a red-bordered entry).

ColorA, by contrast, is composed of three major blocks of code, related in a manner reminiscent of CDD's printer control file structure ([click here](#) to see an annotated copy of ColorA.HTM).

In the first block, a series of local variables are created and assigned (hex) color values. These are the "real" color assignments ([click here](#) to see an annotated copy of this section of ColorA.HTM).

In the second block, a color specified by a skeleton (runtime, by name) is dereferenced to the color assigned in the first block ([click here](#) to see an annotated copy of this section of ColorA.HTM). The third is "simply" for display when you launch ColorA.HTM in a browser ([click here](#) to see an annotated copy of this section of ColorA.HTM). The runtime importance of this last block is emphasized by the

```
omit "1==1"
```

at its beginning. This code is simply ignored at runtime.

In Operation

How this works is somewhat complicated. A skeleton requests a color assignment by making an assignment something like:

```
FinalColor="Cell"
```

which is hard-wired in the skeleton.

ColorA.HTM picks up the value assigned to "Cell." First, the dereferencing block finds the name of Cell's corresponding value in the assignment block. (It also finds the HTML attribute, usually <bgcolor>, to which the color is to be assigned in the final HTML.) This allows a look-up into the assignment block and assembling the result in the generated HTML.

The logic is simple:

$$\begin{array}{l} A = B \\ \underline{B = C} \\ A = C \end{array}$$

And, bingo, the selected color is displayed. Simple, huh?

For example, in WinCore.HTM:

```
<tr finalcolor="Header">
```

declares a row of a table which is to have "Header" as its background color. In ColorA, we find

```
<TSSCRIPT tag="<* FinalColor=Header>" attr=bgcolor  
value="HeaderColor" phase=*>
```

in the dereferencing block. "Header," in this second block, is internally referenced to "HeaderColor." Then, in the assignment block, "HeaderColor" is assigned #a0b8c8 (a light blue). Therefore, this row of the table appears with a light blue background, a sort of cascading set of equates (this example happens to be the title bar,



in the first row of the <TABLE> , by the way).

Changing Colors

One obvious implication of this structure and its implementation is that, should you wish to a change color assignment, you should do so in two places. You should change the assignment in both the assignment and display blocks of skeleton code. Changes in the assignment section will affect your apps. Parallel changes in the display section will let you see what colors are assigned to each variable ([click here](#) to see an annotated copy of ColorA.HTM). And, if you make changes in both places and they are not exactly the same...well, that would get interesting, wouldn't it?

You don't actually *have* to change both, though I do recommend it. If you know what colors you want, you only *need* to change the assignment in the first area. If you are content to look but not touch, changes in the display area need not be accompanied by matching changes in the assignment block (that's sort of pointless, though).

Of course, you also have the option of changing the value a skeleton points to. Group.HTM contains, for example:

```
<tr finalcolor="Header">
```

You could modify Group.HTM so that this reads:

```
<tr finalcolor="CellB">
```

Changes to ColorA affect groups of controls. This method allows changing the color assignment of a single control type.

You could also create your own color assignments:

```
<tr finalcolor="Arnold">
```

All you have to do is define "Arnold" in both the dereference and assignment sections (if you want to see what you've done, the display section also). This process is, more than anything, a matter of copying and pasting.

Who, What And Where?

So what actually controls what?

By the subtle stratagem of changing a value in ColorA.HTM, saving and running an app against it, changing the next value and running again, I was able to determine that the following variables affect backgrounds for the following types of control:

BorderColor	Panels Toolbar Unselected Tab List box cell border
HeaderBColor	List box column headers

HeaderColor	Title Bar Selected Tab Groups
CellBColor	List box cells
CellColor	Sheet (tab background)

Moving through ColorA.HTM, changing colors (select something easily seen; green works quite nicely) and running an app against it is a worthwhile exercise and I recommend it to you.

Background Images

Couldn't a Style Sheet, similar to that created for fonts (["Skeleton Basics: Logos and Fonts"](#)) be used to set background images? Absolutely.

But you don't want to use a Style Sheet for this purpose (trust me on this). You will gain nothing from doing backgrounds this way. In fact, you may well lose some flexibility.

The basic syntax to place a background image on a web page is:

```
<body background="plastic.jpg">
```

And, I have established that the <BODY> tag is in WinCore.HTM. So, it should be possible to nominate an image in WinCore and be done with it.

```
<body background="plastic.jpg" finalcolor="Page"
  bgcolor="white" onload="onBodyLoad()"
onunload="onBodyUnload()">
```

(A Style Sheet wouldn't really have added much, would it?)

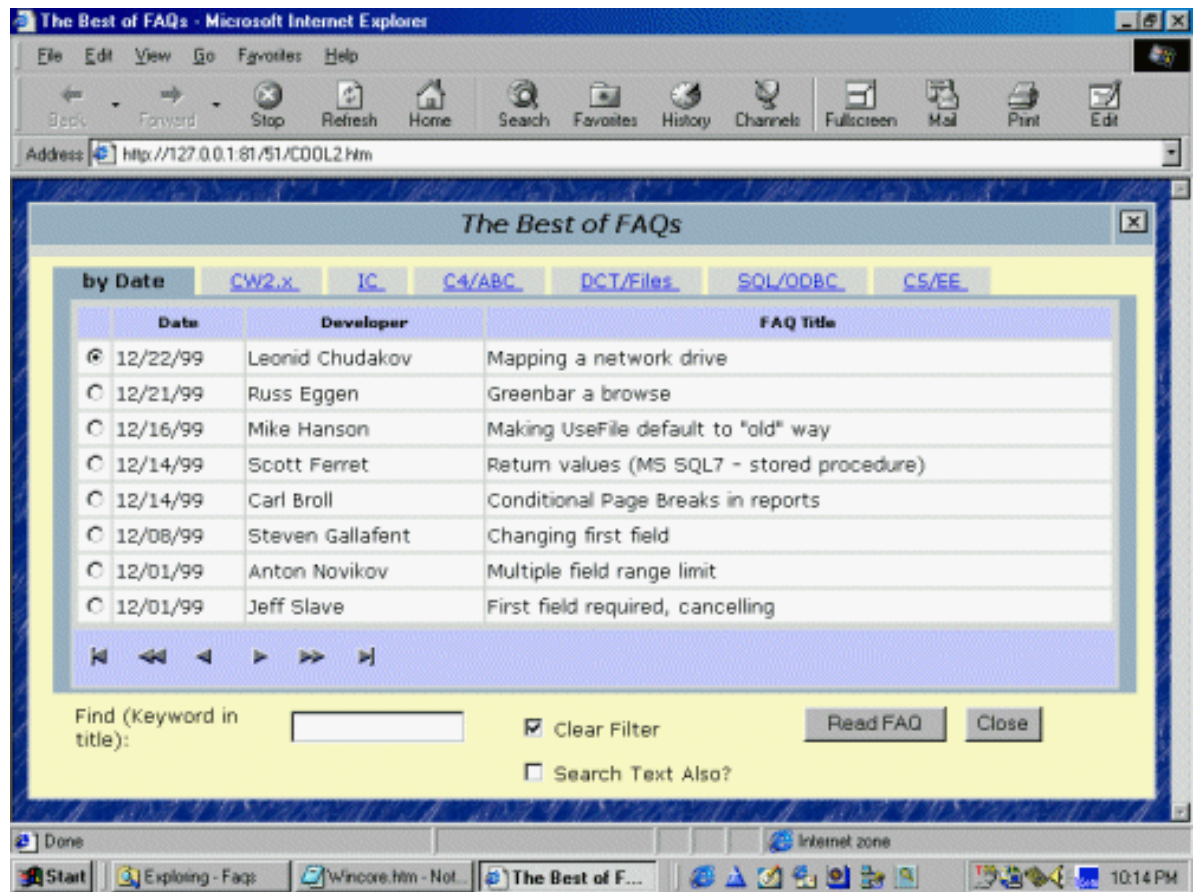
Indeed, if you do so and open WinCore in a browser, the background image appears to work, as shown in Figure 1.

Figure 1. The plastic.jpg background image.



But when you use this modified WinCore.HTM in a real app, you get a somewhat different result (see Figure 2).

Figure 2. The background image is covered by tables.



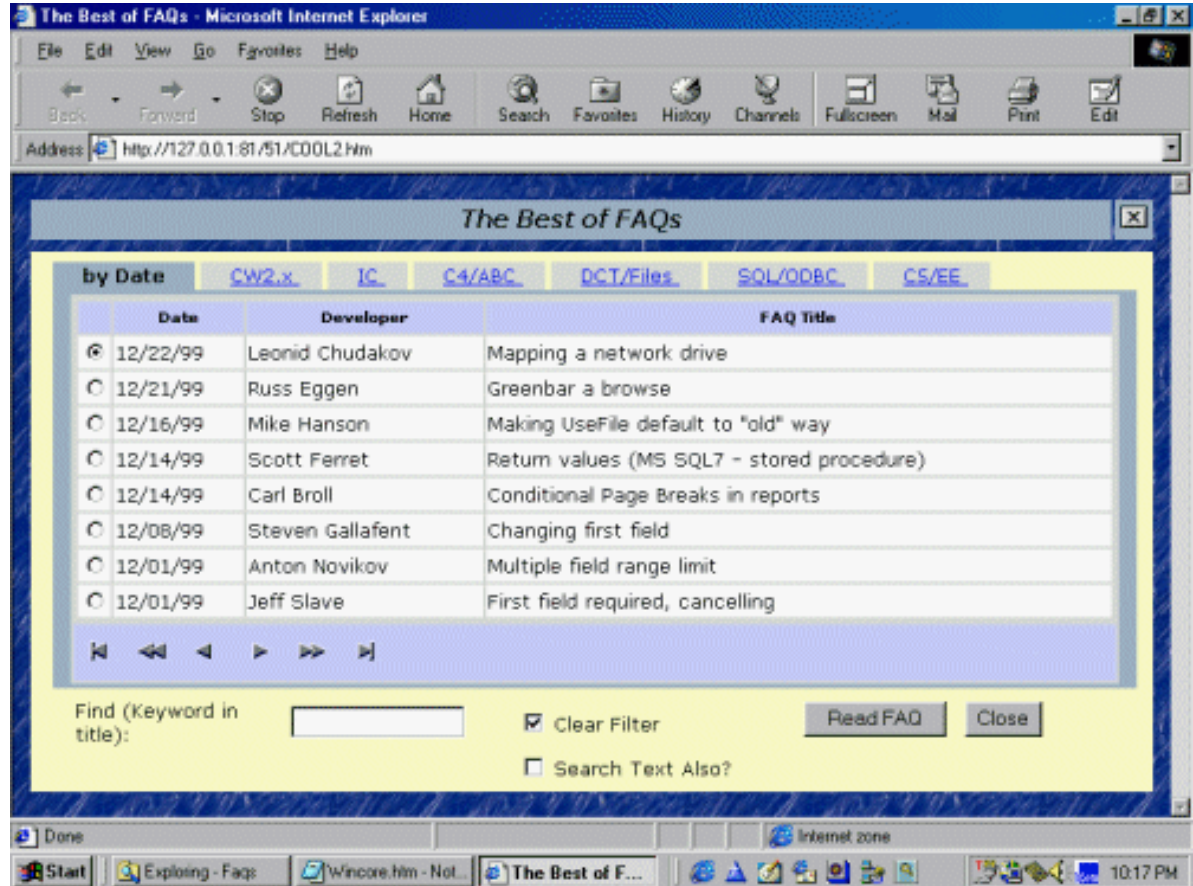
In fact, this is just what one ought expect, though it is hardly what is desired. The HTML for the browse box is in a <TABLE> and that <TABLE> ends up sitting on top of the background. (Note that when you display WinCore.HTM in a browser, there is a box stating "Wizatrons will place controls in here." In fact, there are three similar boxes (menus, toolbars, controls). These are where new <TD>s will be created at runtime, *over*

the background).

To get fuller coverage, there are some additional places at which the image must be named and set.

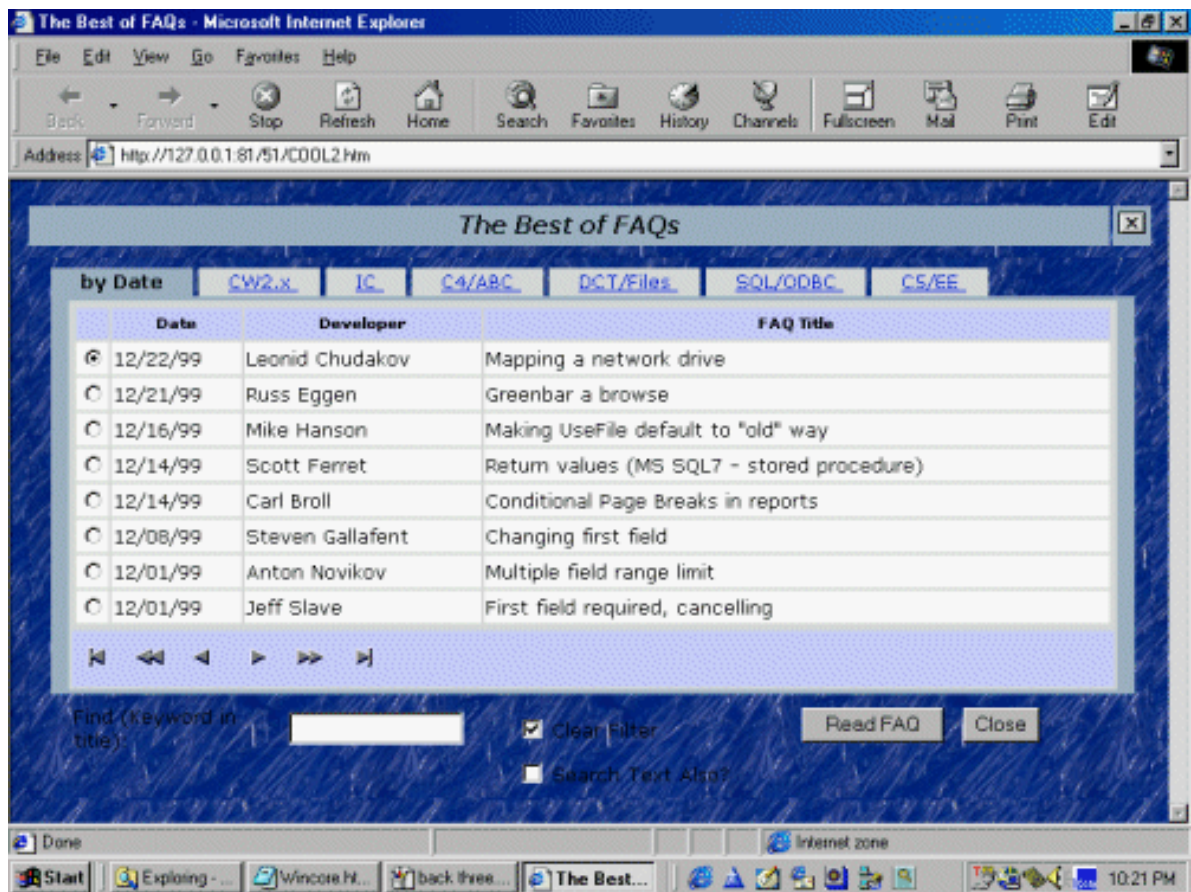
Inside the <TABLE> declaration, covering the pale blue "border" around the title bar and sheet area yields Figure 3.

Figure 3. Replacing the border with the background image.



And, in the second <td> declaration, covering the sheet area (but not the tab) setting the background image. goes the rest of the way, as shown in Figure 4.

Figure 4. Further progress with the background image.



The title bar remains (it is in the first `<td>` declaration in the `<TABLE>`, just before the cell containing `x.gif`). You can delete it, you can nominate the background image in the title's HTML row, you can leave it alone (if you were to use CSS to attach an image to the `<td>` tag, you would lose the ability to choose how you handle this row). My knowledge base, at www.par2.com/cws/c5launch.dll/faqs/coolfaqs.exe.0, places the image into all possible areas, should you want to see a sample. If you look closely, you will see that the image "layers" (the lightness of the image can make this hard to see) instead of tiling from left to right, top to bottom.

The virtue of this is that you may use different images for the page area and the object on it.

NOTE: I am [attaching](#) the WinCore.HTM used to create these figures. There are three places, in this particular implementation, at which I've nominated Plastic.jpg and I have also placed comments to make them a bit easier to find.

And Then Again...

Having to assign a background image multiple times to get a single background seems a bit convoluted. Indeed it is. But the exercise does help get a handle on how the skeletons, especially WinCore, operate. Nevertheless, I can place a `<TABLE>` over an image in an HTML editor without having to name it as the background for both the page and the `<TABLE>`. Why I can't do this here?

Actually, I can (had I been thinking less linearly and more webbishly, it would have been obvious).

WinCore provides a background color for the various tables and the `<TABLE>` elements it creates. Because the `<TABLE>` and its cells have a color, the

background image does not show through. (Where is the TRN attribute when you need it?-Say, why doesn't HTML have a TRN attribute?)

Why not just eliminate the color (which creates the <bgcolor> attribute) assignment entirely?

Why not indeed?

WinCore has two main color assignments that need to be deleted to make <TABLE>s transparent. In the <TABLE> declaration:

```
<table finalcolor="Border" border="0" cellpadding="4"
cellspacing="2" width="100%">
```

and in the second <td> tag:

```
<td finalcolor="Cell" align="center" colspan="2">
```

Doing the same in the first <td> tag will make the title bar transparent. And, of course, you can mix and match these three deletions.

(If you are using a toolbar or menu, you will need to decide whether or not to make them transparent.)

The virtue of this method is that images tile more smoothly. Either way is about the same amount of work.

So, it *is* possible to nominate an image in WinCore once and be done with it (and CSS wouldn't have helped here either).

Summary

Reviewing my previous articles on WebBuilder skeletons, I notice that I use the concept "it's not rocket science" quite a bit. Well, it isn't.

But it finally strikes me why so many of us are having trouble with the skeleton technology. Conceptually simple, it simply isn't linear (even the ABC classes are fundamentally linear). Instead of applying method and logic, we have to experiment to get the effects we're looking for. Trial and error is, to date, the preferred method of "coding" the skeletons and no one worth her/his salt in the development world routinely works that way (well, ok, some of us do occasionally).

On the other hand, because you can get where you want to go by trial and error, you can get used to it (after all, it does work). It is also becoming clear that "prototyping" in an HTML editor and checking the generated HTML is advisable.

Hey, maybe this is the real reason TopSpeed assumes that we'll turn the app over to a web-head.

[Download the source](#)

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

[The Novice's Corner: Clarion Code 3](#)

(Jan 25,2000)

[Skeleton Basics III: Colors and Backgrounds](#)

(Jan 25,2000)

[January 2000 News](#)

(Jan 25,2000)

[The Cranky Programmer](#)

(Jan 25,2000)

Copyright © 1999-2000 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca.

Clarion MAGAZINE

Clarion
Development
Resources

published by

CoveComm Inc.

clarion magazine

Good help isn't that hard to find.**\$6.²⁵/month**[Main Page](#)[Log In](#)
[Subscribe](#)
[Renewals](#)[Frequently Asked
Questions](#)[Site Index](#)
[Article Index](#)
[Author Index](#)
[Links To
Other Sites](#)[Downloads](#)
[Open Source
Project](#)
[Issues in
PDF Format](#)
[Free Software](#)[Advertising](#)[Contact Us](#)

News

Clarion News

January 25, 2000

[New Lower Pricing for List & Label](#)

In recognition of the growing popularity of using List & Label amongst the Clarion community, [Combit](#), the producer of List & Label, has now created a Clarion-only version of their award winning software. This version is the current version of List & Label, but can only be used with Clarion products. It is only available through Solace software as part of the bundle package together with the List & Label templates, and is available for a trial period at £299 + P&P (US\$496) instead of the normal price (if bought separately) of £426 + P&P (US\$707). The only difference between the standard, full Windows Development version of List & Label and the Clarion-only version is that you may not use the Clarion-only version with other languages such as Visual Basic, Java C++ etc. If you purchase the Clarion-only product and later find that you wish to use List & Label in other languages, you may then upgrade to the full Windows Development version of List & Label. The full Windows Development version of List & Label is still available at the normal, discounted price, as a bundle.

[CPCS Emler Now Compatible With Outlook 2000](#)

The CPCS Report Emler add-on has now been updated to support Outlook 2000. There have also been some fixes for previously reported problems. The new install file is available from the CPCS website.

[Sterling Data Clarion Prize Draw](#)

Sterling Data is holding a prize draw worth up to \$195 (depending on the product chosen). All entries must be received by January 30, 2000, and the draw will be on January 31, 2000. All entrants will be notified by email of the winner's name, which will also be posted on the web site.

[PD Date Tools Upgrade](#)

PD International Date and Scheduling Tools is undergoing a rewrite. For a preliminary look and demo see the web site.

[EnTabber 1.2 Update Available](#)

The EnTabber, Enter/Tab solution has been updated to version 1.2. This release fixes a problem under Windows NT 4.0 which sometimes caused an error message during startup. This update is free.

[The Novice's Corner:](#)

[Clarion Code 3](#)

(Jan 25, 2000)

[Skeleton Basics III: Colors
and Backgrounds](#)

(Jan 25, 2000)

[January 2000 News](#)

(Jan 25, 2000)

[The Cranky Programmer](#)

(Jan 25, 2000)

[Read The December 1999 News](#)

January 18, 2000

**How to
Put More
WOW!
into your
apps**

[Click Here](#)

[East Tennessee Clarion Conference & Gathering May 23-27, 2000](#)

The East Tennessee Clarion Conference & Gathering (etc2000) Is On! Scheduled for May 23-27 at the Edgewater Hotel in scenic Gatlinburg, Tennessee, etc2000 looks to build on the success of the 1998 event. Speakers include Dave Harms (Clarion, Linux, and Java: Managing a Multi-Platform Website Environment), Nik Johnson (Use ALL the Clubs in the Bag), Steve Parker and Skip Williams (Thin Clients And Thinner), and Andy "Cowboy" Stapleton (Taking Sybase SQL to the Web). Andrew and Sabrina Guidroz will once again be displaying the art of Cajun cooking as they did in '98 (that pig-in-a-coffin has become the stuff of legend). Programmed activities are also available for guests.

[New TX Control Class Wrapper](#)

A new version of the class wrapper for the TX text processing control is available. New features include support of the new TX 7.0 features such as headers and footers, page numbering, and marked hyperlinks.

[Linder SetupBuilder 3.0 Goes Gold](#)

Linder SetupBuilder 3.0 is now in final release. SetupBuilder is a Rapid Setup Development tool for Windows 95/98 and Windows NT 4.x/2000, with a visual development environment that does not require knowledge of a script language. The system compiles professional, high performance 16- and 32-bit installation programs as a single self-extracting EXE file (or one file per disk for multiple disk installations). The product provides support for Binary Update Patching to distribute repairs and updates. Typically patch files are 10 to 15 percent of the size of the entire set of files. Other features include selective installations, dependency watch, billboards, configuration checking, checking for an expiration date, efficient data compression, Wise Script Import, Windows 95 and Windows 2000 Dialog Box Style, and more (many more!). Installation program overhead is 90k. Linder SetupBuilder costs \$119.00 for a royalty-free usage license. A trial version is available.

January 11, 2000

[NetTalk 1.0 Beta 3 Released](#)

The first public beta of NetTalk is now available. Although not yet feature complete, it is usable in LAN environments. WAN support is expected soon. NetTalk is a toolset that allows your programs to communicate with each other over TCP/IP networks. The focus has been on ease of use for the developer, and simplicity of deployment. Although the emphasis has been on allowing any kind of data to be exchanged, NetTalk comes with prebuilt objects file transfer, closing apps remotely, workstation time synchronization, and chat. Introductory pricing during the beta program is \$199, going up to \$299 at release.

[List & Label Templates Version 1.6r Released](#)

Version 1.6r of Simon Burrows' popular List & Label Templates is now available. New features include invoice type reports, the ability to modify List & Label options/behaviour, date fields with variable offset values, multi-language support, direct report calls (rather than requiring selection from a list), a hand code template, and more. A demo application is available, and a support page is also available at the website.

[CapeSoft Secwin 3.0 For C5.5 Ships](#)

CapeSoft has released SecWin 3.0 for Clarion 5.5. Features include complete login



/ password functionality as well as individual screen and control protection., product registration functionality with including control over the end user installation through the use of Activation Codes, and an easy-to-use interface. Alternative driver support includes TopSpeed, Btrieve, ODBC and MSSQL (more on the way). Secwin 3.0 supports Legacy and ABC, 16 and 32 bit, local and standalone compiles and any combination of multi-DLL and multi-EXE applications. The product is available for Clarion versions 4 through 5.5. As in the past the 16 bit DLL version of Secwin is fully functional and completely free. The Registered version is available at the reduced price of \$79 (normally \$99) until Jan 31, 2000. There is no charge for upgrading from an earlier version.

[TearOff Special Ends Soon](#)

CapeSoft's TearOff toolbar product is on sale through the end of the month for \$29 (reg price \$39). Orders can be placed at www.clarionshop.com.

January 4, 2000

[ClarionNet To Join Accessories Program](#)

ClarionNet templates and libraries make it possible to web-enable new and existing Clarion programs without using HTML. ClarionNet allows the remote rendering and usage of Clarion programs, with some limitations. Applications are deployed via the Clarion Application Broker.

[New Demo Web Application](#)

TopSpeed has a new Clarion 5.5 web-enabled demo app available for download. This application can run from TopSpeed's web site, or downloaded and run locally with a browser (using the "linked-in Application Server"). Source is also available for those who have Clarion 5.5 beta 1.

[CapeSoft TearOff 1.0 Released](#)

CapeSoft's TearOff is a small inexpensive template that adds a dockable toolbox to your applications. Users can easily create their own dynamic toolbox from existing menu items. On Special until 31 January 2000 for \$29 (regular price \$39).

[Imaging Templates Version 1.08 Now Available](#)

Imaging Templates Version 1.08 is now available for download by registered users. This release contains some minor fixes to clipboard functionality and image deletion. Local apps are not supported for 5.5 (apparently a Clarion beta issue) but standalone apps work fine.

[John Herron Forms Kimarx Technology Group](#)

John Herron, formerly of TopSpeed Corporation, has announced the formation of Kimarx Technology Group. The mission of Kimarx is to provide high quality Clarion (and non-Clarion) talent at a reasonable price. The company offers an array of services including extra manpower, system analysis, specifications, project management, documentation, and testing. Kimarx is looking for developers, clients, and development partners. 606-647-6656 or 606-647-6636 (fax)

[The \\$99 IFT Sale Returns](#)

IFT:HTTP Server 2.0 is back on sale at \$99. The Internet Framework Template HTTP Server edition gives you the power to build your own, custom, 32-bit Windows, HTTP server application using Clarion. The Internet Framework Templates handles the interface to the Windows socket (Winsock) functions. Version 1.5 upgrade is free to current IFT customers. Free PowerMerge Mail Merge Beta Templates included (single

item and inline list search and replace.

[Data Modeller 5.5 Announcement](#)

DM 5.5 is now out of beta testing and is available as an upgrade for DM 5.0 users. Prospective users of Clarion 5.5 will receive this new and existing DM as part of the Clarion 5.5 Enterprise Edition. This new DM 5.5 will have many benefits including for the first time the ability to plan your application during the design phase.

[Send Internet Mail 3.04 Released!](#)

Send Internet Mail 3.04 has been released. New features include updated documentation, enhanced demos, and SetFrom accepts both name and address. This release also includes several bug fixes.

[XLIB Holidays Discounts](#)

Through January 7, 2000 XLIB is available for US\$200 with full source (25% discount) or for US\$48 with INC and LIB files only (20% discount).

[New CPCS Examples Posted](#)

Larry Teames has posted several new CPCS example applications, showing how to provide user-selectable multi-sequence reporting, and how to create and use overflow headers. Existing users of CPCS v5.15 may download and review these examples.

Copyright © 1999-2000 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca.

published by
CoveComm Inc.

clarion magazine

Good help isn't that hard to find.**\$6.25/month**[Main Page](#)[Log In](#)
[Subscribe](#)
[Renewals](#)[Frequently Asked
Questions](#)[Site Index](#)
[Article Index](#)
[Author Index](#)
[Links To
Other Sites](#)[Downloads](#)
[Open Source
Project](#)
[Issues in
PDF Format](#)
[Free Software](#)[Advertising](#)[Contact Us](#)

Opinion

January, 2000

The Cranky Programmer

Comments On Comments

Well, here we go...into the new mil-looney-um.

I waited for a couple weeks to see if anything had changed now that it is (supply your own suitably portentous and echo-y voice here)... "THE YEAR 2000!"

Nope.

Still cranky.

Maybe it's because this is the faux millennium (the real new one doesn't actually start until 2001), but it seems like everything is just the same-old-same-old hanging around.

And speaking of hanging around, one thing Y2K should have taught us all is that the code you write can have a lifetime *far* beyond what you expect to. I mean, I've just spent half the morning doing support for a CFD system (that's Clarion For DOS, for all you young 'uns) that I thought went away a couple of years ago. I've also inherited a large system of CW 2.0 code that had been expanded, enhanced, abused, chopped, sliced, diced, tortured and basically stretched completely out of its original shape over a number of years.

And that brings me to my point (you knew there had to be one in here somewhere, didn't you? I mean, didn't you?): Comments. As in commenting program code.

Everyone knows the truism that program code can become completely mysterious even to the author after a period of time, so I won't repeat it here. (Wait, I just did...dang!) I mean, is there *anyone* among us who hasn't gone back to an old piece of code and stared at it while wondering "What the *&#! was I *thinking* when I wrote that?" Be honest!

Now multiply that mysteriousness by tossing in a few random facts such as: you didn't write the code in question; there is no system documentation; and there is no one to guru you through it.

And now *you* are now supposed to maintain and enhance that code. (Don't think this won't happen to you at some point – it will!)

The result of all this is that, no matter who wrote the code, a lot of time is wasted learning or re-learning something that should have been a no-brainer (or at least a mini-brainer).

The simplest solution to this type of problem (and the cheapest time-wise), is to make sure your code is both easy to read and self-explanatory *the first time through*. (SURE you'll get back to

[The Novice's Corner:
Clarion Code 3](#)
(Jan 25,2000)[Skeleton Basics III: Colors
and Backgrounds](#)
(Jan 25,2000)[January 2000 News](#)
(Jan 25,2000)[The Cranky Programmer](#)
(Jan 25,2000)



it. SURE you'll take care of it next time you look at this section of code. SURE the sun will rise in the west tomorrow!)

Easier said than done, it seems, as in my travels I have observed quite a wide variation in coding and commenting styles. I'd like to share a few of them with you, followed by a few of my own (obviously *perfect*) commenting and coding style rules.

The Code Watchers Guide To Commenting

To begin at the beginning, there is...

The 'No-Quiche-For-Me' Crowd

These are people who live by the credo "*Real* coders don't comment their code!" They would sooner put on a fuzzy mouse suit and wrestle a barrel full of hungry cats than let anyone else know what they were thinking (after all, the brilliance of their code should be obvious to everyone, right?).

Here's a very simple example:

```
If Tax > Max
    Tax = Max
End
```

So, in this case, we can see fairly easily *what* is happening. But can you tell me *why* it is happening? Hmmm? That's right, now you spend the next half-hour trying to find out all the rules for Tax and Max and why we might want to do this. And this is only two lines of code (the "End" doesn't count).

This style (or more precisely, *non*-style) is pretty bad, but wait until it is coupled with...

The Wheel Of Fortune Contestant

The adherents to this style couple a lack of comments with a rather miserly attitude, as if every single character they use costs them money ("Well, if you're going to *force* me, I guess I'll buy a vowel, Pat").

This type of code abounds with single letter variables, occasionally (and grudgingly) expanded to two letters merely because there are *only* twenty-six measly letters in the alphabet. So you might see:

```
If T > M
    M = T
End
```

Same basic dilemma as before, only this time you don't even have the benefit of the variable name to guide you. What does T mean? Or M?

Actually, I'm even being a bit too obvious here, what with all those extra line feeds and spaces. For a true disciple of this method, the code would look like:

```
If T>M;M=T.
```

Oh, yeah! *Now* we're cooking. The embed points look like someone simply poured in a can of alphabet soup, stirred vigorously and then *dared* the compiler to make sense of it. ("Double, double, toil and trouble"... for you, that is.)

Masters Of The Obvious

Here we move to the opposite style of commenting, namely someone who comments *everything*. The problem with this particular style, though, is that many adherents simply echo



what the code says. For example:

```
If T > M      ! If T greater than M
    T = M      ! Assign M to T
End           ! If T > M
```

While it looks superficially appealing, the end result is that the code is cluttered with a ton of extra text that really doesn't tell you much of anything. More specifically, it doesn't tell you *why* you would want (or need) to assign M to T if T is bigger.

The Mumbler

When reading this type of comment, you really get the feeling that the person was just sort of mumbling into the keyboard. These comments are rambling, sporadic, and have an annoying tendency to not be relevant to the code.

Mumblers can also come up with some really scary comments, such as this little nugget (and this is from real production code in a commercial product):

```
ReChar# = 1 ! I knew this would come back to haunt me...
```

Gives you *lots* of confidence in the code, doesn't it?

Michael Meyers Syndrome

Ah, yes. These are the people who rampage through the code, chopping out multiple lines and sections using '!' or omits – without bothering to say why they did it. Why *was* that code bypassed? Was it important? Why was it there in the first place? They'll never tell. And it will take you forever to figure it out.

The "I-Told-You-Once" Crowd

These types will comment the code initially... and then never update the comments again. Here is where you see gems like:

```
If Tax > Max ! Check for too much tax
    Tax = Max + SRV ! Subtract the overage
End
```

Want to take bets on this code?

I thought not.

The Novelist

This admittedly rare specimen falls into the too-much-of-a-good-thing category. There are copious notes. Volumes of comments. So many notes and comments, in fact, that you can't see the code itself. While preferable to most of the other styles, it still takes more work than it should to wade through it. (I sometimes think these are people who learned to program while being paid by the line.)

Ok, I could go on and on, but let's cut to the chase.

The Cranky Programmer's Simple Rules of Code Salvation

I try to live by four simple coding rules:

1. Use readable, explanatory variable names. If you are doing something with a maximum tax amount, call it "MaximumTaxAmount", not "MTA". A bit of extra typing now will pay off big time later on. And when dealing with logical options, use the equates provided by the system, such as True instead of 1 and False instead of 0. Equates are

your friends!

2. Break the code into logical sections, using white space to keep it readable and easy to scan (i.e., with blank lines and code alignment). Half the effort of finding a problem is locating the lines in question. By making it easy for your eye to grasp what is going on in each logical block, you will locate the source of (and for) the problem with much less effort.
3. Where needed, preface the logical sections with an explanation of *why* the code is there. Don't describe the code itself, describe the reason it is in the program in the first place. And if you comment out some code, explain why you did it. My personal preference is to place comment/header lines after a blank line, and set them off with a '!---' or similar (see below). It really makes comments stand out, and provides logical organization for what I'm doing. For example, a good test is to imagine what the embed or source block would look like with *only* the comments. If done right, you end up with a logical outline which describes exactly what you are trying to accomplish in that section of the program. Gee, that almost sounds like documentation to me.
4. Finally, the kicker: DO IT NOW! This is along the lines of my earlier comments, but its importance simply cannot be overstated. You'll *never* know as much about the what and why of a section of code as when you just wrote it. It's all there in your mind – the logic, the reasoning and the big picture. And it *won't* all be there the next time you come back to this code; something else will have pushed it out and/or muddled it up.

There's one more thing which can make code harder to scan. This is when a logical test is continued onto another line in a non-obvious way. Consider these two examples:

```
!---Example 1 (sure to incur the wrath of the Cranky Programmer)
If Something = AnotherThing or |
DoSomething = True DoNothing = False End

!---Example 2 (gets you the gold star on your forehead)
If Something = AnotherThing |
  or DoSomething = True
  DoNothing = True
End
```

In scanning the first example, it is very easy to miss that trailing "or" statement, and to thus assume that `DoSomething = True` is an assignment. In the second example, it's obvious that it is really a comparison and is part of the "If" statement.

Tidying Up

Of course, there a million other possible rules about indenting, variable naming, code organization and so forth, but the bottom line is that you want the code to be easy to read, broken into logical segments, and most of all, self-explanatory.

Trust me, even if you don't do it for yourself, do it for the poor schmuck who might inherit your code someday.

Or, on some dark and stormy night, you just might see a crazed programmer with a vengeful grin lurking outside your door...

So, Got Any Comments?

Drop me a line, get it off your chest. Ask me a question – who knows, maybe I'll answer it in the column. Come on, here's your chance to get cranky and help the world at the same time.

I remain,

cranky@clarionmag.com

[The Novice's Corner: Clarion Code 3](#)

(Jan 25, 2000)

[Skeleton Basics III: Colors and Backgrounds](#)

(Jan 25, 2000)

[January 2000 News](#)

(Jan 25, 2000)

[The Cranky Programmer](#)

(Jan 25, 2000)

Copyright © 1999-2000 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited. If you find this page on a site other than www.clarionmag.com, email covecomm@mbnet.mb.ca.