Clarion Magazine

This edition includes all articles, news items and blog posts from January 1 2011 to January 31 2011.

Clarion News

Read 20 Clarion news items.

The ClarionMag Blog

Read 5 blog entries.

Articles

ClarionMag Tip of The Week #6: Formatting Text The Way You Like It

January 6 2011

Probably the nicest thing you can say about the old Clarion editor is that it's familiar. But in just about every other way the Clarion 7 editor is a massive improvement. Dave Harms lists his favorite editor formatting features.

MagGem: Gender Guessing and Regular Expressions

January 12 2011

Guessing a person's gender from their first name isn't the usual sort of Clarion fodder, but it turns out to be a nice introduction to the power of regular expressions.

Product Review: NetTalk 5, Part 1

January 13 2011

NetTalk 5 is CapeSoft's flagship product for web development. While that often means browse/form business apps (stay tuned for Part 2), there are a lot of other things you can do with NetTalk, from email and FTP to inter-application communication across the LAN.

Clarion Tip of the Week #7: Changing Editor Font Size On The Fly

January 14 2011

Yes, monitors are getting bigger, but the default font in the C7 text editor is still a mere 10 points. You can change it in Tools|Options, but there's an even easier way.

MagGem: A String Class

January 15 2011

Yes, we have a bunch of string handling functions in Clarion. But sometimes a string class is just so much better.

MagGem: SQLite Is Tiny And Useful

January 18 2011

SQLite isn't the answer to every SQL problem. Really it's only the answer to a few. But it has some hidden talents.

MagGem: Let Your Hamster Do The Work

January 21 2011

SoftVelocity is still sorting through the cause of painfully slow newsgroup access. Want to avoid the hassle? Leave everything up to Hamster.

Tip of the Week #8: Side By Side Windows

January 27 2011

Sure, you can open multiple apps at the same time in C7, and dictionaries and other files as well. But that's just the beginning - you can also also view those windows side by side.

MagGem: Reflection in Clarion

January 27 2011

Okay, it's not true reflection like you have in .NET, but you can get a whole lot done with Who(), What() and Where().

Creating an SQL Query Class and Template, Part 1

January 28 2011

Phil Will takes an SQL query procedure created by Bob Huff and refactors it into a class and a control template. In Part 1 Phil creates a class shell and a bare bones template.

Creating an SQL Query Class and Template, Part 2

January 31 2011

In this second of two parts 2 Phil simplifies the code needed to handle a large number of controls and their events.

Clarion News

PD Class Generator Version 7.3.0005

An updated version of PD Class Generator is now available. It includes several enhancements among them a new useful example, a SQL Query Control Template and Class based on a procedure developed by Bob Huff and shown on ClarionLive!. The control allows users to develop and test queries in an SQL environment.

Posted January 7 2011 (permanent link)

Clarion 7.3.7900 Released

SoftVelocity has released Clarion 7.3 build 7900. This is the first public C7.3 release, with numerous fixes and improvements from C7.2.

Posted January 14 2011 (permanent link)

SetupBuilder 7.3 Build 3195

SetupBuilder 7.3 Build 3195 is available, free of charge, to all SetupBuilder customers who have an active SetupBuilder maintenance and support plan subscription.

Posted January 14 2011 (permanent link)

ComSoft Products Updated For 7.3

Installers for ComSoft's products have been updated for version 7.3. They are all source code, so do not need compiling. Products include: BoTpl Free; BST; BFlat; BSec; BRT; BoTran; BoTran2; BST 4.26 (new embeds added); BATT; BBTT; BHT; BPL; BMT; BFET; BIT. BSRef has not yet been updated, as it has other features currently being added.

Posted January 14 2011 (permanent link)

CPCS/RPM C7.3 Installs

Up to date installs for Report and Presentation Manager are now available for Clarion 7.3. Creative Report installs should be available on the CPCS site shortly. This release of RPM uses the same libraries for ABC and Legacy template families. This change increases the variable lengths of several older option paths and uses external INC files for the option group definitions.

Posted January 14 2011 (permanent link)

Clarion FreeImage For C7.3

The Clarion FreeImage installer has been updated to detect and support Clarion version 7.3. The only change is the installer.

Posted January 14 2011 (permanent link)

iQ Products Updated For C7.3.7900

All iQ products have been built using build C7.3.7900. These include: IQ-Notes Version 5.04; IQ-Sync Version 1.35 (includes fixes to subsequent copies over read-only file.); IQ-XML Version 2.73. If you download iQ-XML, make sure you run the IQRecent.APP to Tweak your Clarion 7.3 UI.

Posted January 14 2011 (permanent link)

Solid Software Updates For C7.3

The following products have been updated for Clarion 7.3 and are available for download immediately: RichReport, ImageEx,SysList and SysTree. The installation passwords have not changed.

Posted January 14 2011 (permanent link)

LANSRAD Updates For C7.3

The following LANSRAD products have been updated and tested with Clarion 7.3 and new installers are available for immediate download: ProImage; ProScan; ProCodeBlock and ProPath. Existing customers can use the same download links as previously. Note that ProPath is an upgrade to version 1.5 (even though the filename in the old link still says 1.4) and that it has extended support for multi-DLL apps and several new example apps for multi-DLL setup. ProScan and ProImage users will need to download the update to ImageEx.

Posted January 14 2011 (permanent link)

Fomin Report Builder For C7.3

The Fomin Report Builder build for 7.3 is ready and available for download.

Posted January 14 2011 (permanent link)

StrategyOnline Updates For C7.3

Eleven of StrategyOnline's products are now C7.3 ready.

Posted January 14 2011 (permanent link)

ARD Report for C7.3

ARD Reporter for C7.3 binaries are now available.

Posted January 14 2011 (permanent link)

Smart-Type for C7.3

The Smart-Type installer is now C7.3 compatible

Posted January 14 2011 (permanent link)

Super Templates for C7.3

BoxSoft has released versions of Super Templates compatible with Clarion 7.3,

Posted January 14 2011 (permanent link)

CPCS C7.3 Installs

Creative Report installs are available for download on the CPCS site.

Posted January 21 2011 (permanent link)

Super QuickBooks-Export Templates For C7.3

The Super QuickBooks-Export Templates are now compatible with Clarion 7.3.

Posted January 21 2011 (permanent link)

amazingGUI 1.2.3.1

amazingGUI version 1.2.3.1 has been released. This version includes an installer for Clarion 7.3.

Posted January 21 2011 (permanent link)

DMC 2.3.3.1273

DMC GOLD Version 2.3.3.1273 is now available, with numerous changes and improvements. This version of DMC introduces a new Clarion 7 IDE integration link under Help/Accessory.

Posted January 21 2011 (permanent link)

PD Class Generator Version 7.3.0006

An updated version of PD Class Generator is now available. A new feature allows you to embed multiple blocks of template code within class virtual methods at PRIORITY locations you specify. The #AT statements for the methods are generated for you. The Clarion 7 version has an updated SQL Query Control example - the result of doing some work on an article for Clarion Magazine. There is one minor fix - PRIVATE attributes were not being generated.

Posted January 21 2011 (permanent link)

ClarionTest 1.0.0 Released

ClarionTest 1.0.0 is now available for download, as a cleaned build or as built with C7.3.7900. This release fixes a bug in 0.3 with DLL initialization.

Posted January 28 2011 (permanent link)

The ClarionMag Blog

Jan 4 outage

On January 4 our service provider had a power outage at the data center where ClarionMag is hosted. We were down for several hours - our alternate server was not available as we're changing backup server hosts.

My apologies for the interruption in service. We hope to have the backup server reconfigured soon.

Posted January 4 2011 (permanent link)

C7 addin possibilities

Brahn Partridge has an article coming up on creating addins for C7, in which he shows how to create an addin that lets you search Clarion Magazine from within the C7 IDE!

The basic process of creating an addin is pretty simple, and the possibilities for IDE enhancement are immense.

Brahn already has a bunch of addins available for download from his ClarionEdge site, along with some other useful resources.

Posted January 18 2011 (permanent link)

SV news server requiring logon

You may have noticed that the SoftVelocity news server has been dog slow lately. If you're having problems getting access please note that access is temporarily restricted to authenticated users. Here's what Bob Z had to say (in a newsgroup post):

> So far neither we, nor the news server techs can isolate what's dragging the server down at various points of time. So in an effort to try and isolate/fix this, we are going to turn off Non-Authenticated NNTP connections. What this means is simply that if connecting with an NNTP client program (Outlook, Thunderbid, etc) you will need to enter in your login information (same info you would enter if you used the Discussion client software). While this move won't necessarily immediately fix the problem, it can better help us to identify who is on the other end of the connections that are fouling up the server.
>
> Hopefully we can get this resolved fairly quickly.

Posted January 19 2011 (permanent link)

SV news server on the move

SoftVelocity is moving the newsgroups to a new provider. From a blog post by Z:

> Over the next 24 hours we'll be moving the newsgroups to a new provider, we'll have more bandwidth and a better server, but with the move will come a new IP address, so depending on how fast your own provider gets DNS updates into their system, you may have a short period where your client news reader software may not resolve to the new IP address. So don't panic if it happens that you can't connect for a short period, we use a top-level DNS provider so it should propogate very quickly. In order to ensure that no messages are lost (left behind on the old server) during the final transition to the new server we'll take the existing server offline for a short period for a full backup and deployment to the new server.

Posted January 26 2011 (permanent link)

TPS corruption and Microsoft Security Essentials - again!

Here's a heads up from Robert Paresi. As you've probably heard by now, Microsoft Security Essentials causes corruptions in TPS files, and the only real solution seems to be to remove MSE.

If you're detecting MSE programmatically and warning your users to remove it, be aware that Microsoft changed the program name and registry key. Here, with Robert's permission, is some code you can use to detect the new version:

```
if GetReg(REG_LOCAL_MACHINE,'SOFTWARE\Microsoft\Microsoft Security Essentials','Market') <> '' OR |
GetReg(REG_LOCAL_MACHINE,'SOFTWARE\Microsoft\Microsoft Antimalware','InstallLocation')
message('You cannot use this program with TPS file database.','Microsoft Security Essentials Installed',icon:exclamation)
do procedureRETURN

end
```

Posted January 31 2011 (permanent link)

ClarionMag Tip of The Week #6: Formatting Text The Way You Like It

By Dave Harms

Posted January 6 2011

One of my favorite things about Clarion 7 is the source editor. I particularly like it because it isn't the C6 editor. As productive as I've been with Clarion over the years, the old source editor has been a constant source of torture, especially because of the single-level undo capability. This especially plagued me when working on laptops. More than once while hammering out some code I let my thumb drag over the touchpad while holding the shift key. Result: I selected a bunch of newly-typed text, and obliterated it forever with the second unstoppable keystroke that followed.

Clarion 7, on the other hand, sports a very capable source editor, presumably inherited from SharpDevelop. Not only does it have seemingly infinite undo capabilities (which I've tested on numerous occasions), but it offers some very nice text-formatting options. I'll show how to do these via the context menu, but as you'll note from the Figure 1 there are keyboard shortcuts for many of these options as well.

The Format menu applies to selected text, so go ahead and select some and right-click, then choose Format (Figure 1).

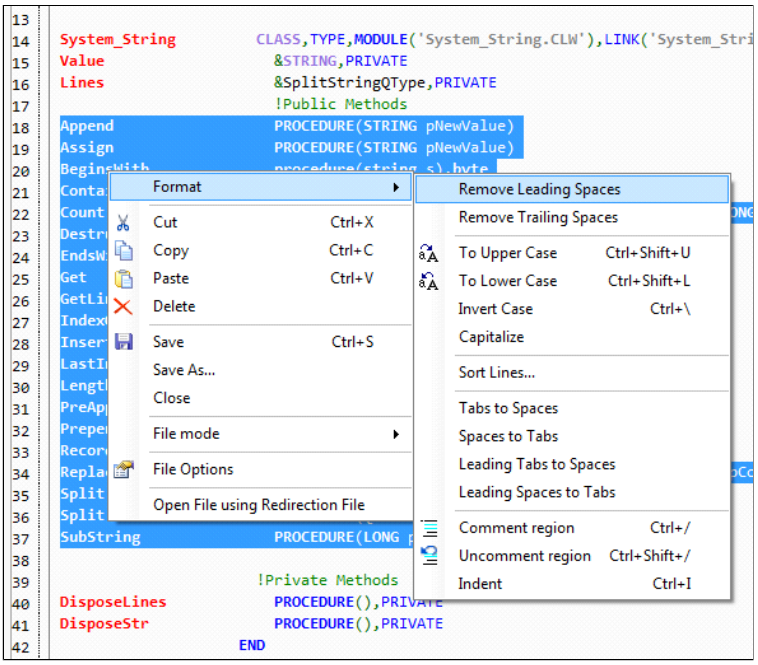

Figure 1. The Format menu

Tabs and spaces

There are six menu items that deal with tabs and spaces.

The first two, which remove leading and trailing spaces, probably aren't going to be that useful unless you're working on cleaning up some text that just happens to have unnecessary leading or tailing space.

The other four, a little lower down, let you switch between tabs and spaces. That functionality may not mean a lot to you, and it doesn't to me, but religious wars have been fought over whether to use tabs or spaces to format code. So if some infidel provides you with that horror of horrors, space-indented code, you can freely switch it to tabs with this, and vice versa.

Case

The are four options for dealing with text case. I'm not a fan of all caps IN THE LEAST, so you won't see me hitting Ctrl-Shift-U a lot, but if you're a true blue Clarionite with a letter of commendation from Bruce Barrington himself you might want to use this to clean up your Clarion keywords.

I'm more likely to hit Ctrl-Shift-L to lower-case everthing, or maybe Capitalize. Actually Capitalize only affects the first character of the word, so if that word is already in all caps it's going to stay that way - Capitalize doesn't lowercase the rest of the letters. If you want that you'll need to make the text lower case first. If like me you have a serious aversion to text in all caps (DO I MAKE MYSELF CLEAR?) then lower casing and capitalizing will get

you well on the way to readability. Just remember that this will destroy any mixed case or camel case naming.

Invert case was, I believe, added after someone who missed the change case functionality in the old editor wrote a plugin to add it to C7.

Sorting

The ability to sort text is pretty cool, especially when it comes to cleaning up class declarations. I like to have my class variables in alphabetical order, and my class methods in alphabetical order.

Unfortunately, while it's easy to do this for the one-line declarations inside the Class structure, there's no way to do this for the actual method code. The best you can do with the Sort menu option is sort all your method code alphabetically, which I do not recommend. Although that would make for an interesting Clarion Challenge. Hmmm.

Commenting/Uncommenting

I use the Ctrl-/ and Ctrl-Shift-/ shortcuts constantly to comment and uncomment code. Yes, that functionality is in C6 but the shortcuts are clunkier - you have to use Alt-E to bring up the Edit menu, then press M to mark and N to unmark.

Indenting

Indenting is probably my favorite editor feature. It's not that great in data sections (where, in my experience, it's wildly inconsistent about spacing and often places the data types way too far to the right), but it's fantastic in code sections. It's much easier than highlighting text and hitting tab and having that strange "how many spaces would you like to tab and in which direction" dialog pop up. Just select the text in question, press Ctrl-I, and the editor will almost always indent your code correctly.

Let me know your favorite editor features.

David Harms is an independent software developer and the editor and publisher of Clarion Magazine. He is also co-author with Ross Santos of Developing Clarion for Windows Applications, published by SAMS (1995), and has written or co-written several Java books. David is a member of the American Society of Journalists and Authors (ASJA).

Article comments

---

by Russell Eggen on January 8 2011 (comment link)

> Dave,
>
> Here is another nice feature of the editor. Sometimes I want to open a file referenced in a module. For example:

INCLUDE('SomeClass.inc'),ONCE

I highlight SomeClass and then right-click. Choose "Open File Using Redirection File". The file then opens in a new tab. Hover the mouse over its tab if you wish to see the path where the file lives.

Is it where you expected to find it?

Russ

---

by Dave Harms on January 8 2011 (comment link)

Good one, Russ! I just tried that with a file that's in two locations on my system, and it correctly opened the one that came first in the RED file.

BACK TO TOP

MagGem: Gender Guessing and Regular Expressions

Posted January 12 2011

Back in 2002 Geoff Robinson wrote a two-parter on what might seem like an odd topic: code that guesses a person's gender from their first name.

Now, there may not be that many situations where you'd want to trust something like this to a computer. Direct mail/email might be one - perhaps you've purchased a list and you will get better results by targeting your material at men or women.

But even if you can't see an obvious use for gender guessing, Geoff's article is well worth a look for a couple of reasons. One is that he discusses regular expressions (aka regex) in some detail. Regex isn't that commonly used in the Clarion world, but support is there in the language via `Match()` and `StrPos()`. If you do any kind of string manipulation, there's a good chance that some of your code would benefit from using a regex.

Another good reason to read the article is that Geoff is a skilled developer; even if you're not doing anything remotely like regular expressions you can probably learn a few things from his lean and efficient code.

Read the article

Watch the MagGem during ClarionLive webinar #86

Article comments

BACK TO TOP

Product Review: NetTalk 5, Part 1

By Dave Harms

Posted January 13 2011

NetTalk 5 is large, complex and powerful Clarion third party product. NetTalk doesn't just "talk" to the Internet or any TCP/IP network, it also lets you build programs that access and deliver all sorts of network services.

NetTalk's features include:

- Email client (SMTP and POP3)
- Web client and server (HTTP and HTTPS)
- Full web application development using NetWebServer templates
- FTP client and server
- Time client and server (for time synchronization)
- Real-time chat
- Remotely close down applications
- Remote application refresh
- Dial-up networking client
- SOAP web services client and server (together with xFiles)
- Dynamic IP management
- SNMP (Simple Network Management Protocol) client and server
- News server client (NNTP)

And since NetTalk gives you the basic tools to manage TCP/IP communication you can use it as a building block to create just about any other network service that comes to mind.

Still, it's possible to divide NetTalk's functionality into two general areas. One is creating Clarion-style business web applications; the other is everything else. In this article I'll cover the "everything else" aspect (which is generally unchanged from NetTalk 4). I'll look at web applications (and the improvements in NetTalk 5) in Part 2, coming soon.

Installation

I downloaded NetTalk from the CapeSoft Accessories page. All CapeSoft third party products are delivered as SAF files, which can only be read by CapeSoft's Safe Reader

utility. You enter the Safe Key you received when you purchased the product, press Tab, and if the key is good you're ready to run the installer (Figure 1).

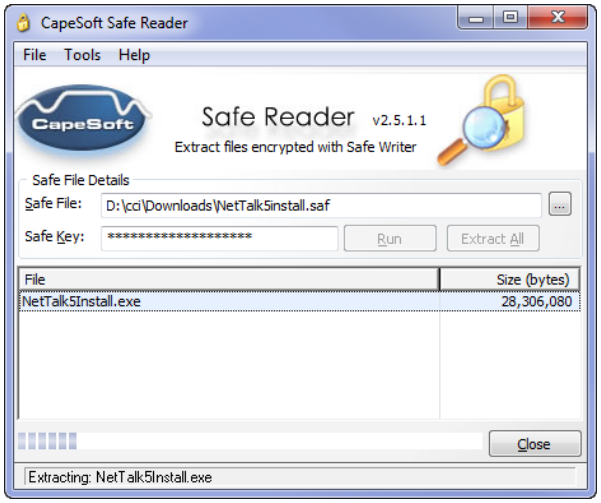

Figure 1. Extracting the installer

On running the installer I discovered that it wasn't detecting Clarion 7.

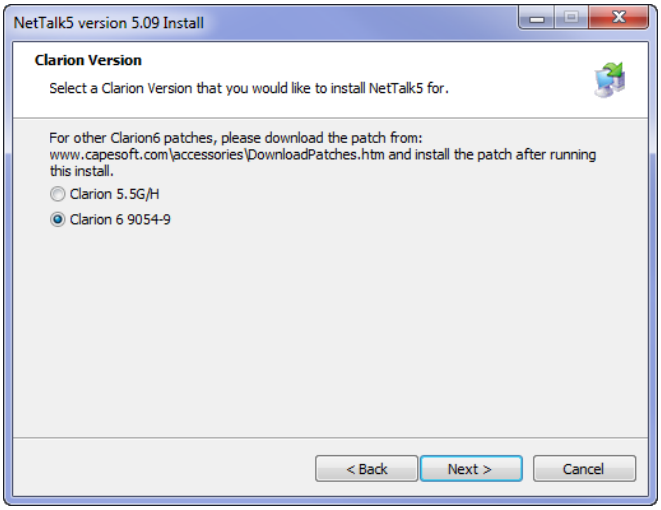

Figure 2. Running the C5.5/C6 installer

Carelessly, I'd simply downloaded using the link in the left hand column when I should've used the link in the right hand column. If I had scrolled up to the top of the page I would have seen that the left column is for C5.5 and C6, and the right column is for C7.

NetTalk 5 (paid for upgrade)	22 Dec 2010	5.09	28 MB Mirror	Patches	29 MB Mirror
StringTheory	21 Dec 2010	1.30	1.5 MB Mirror		1.5 MB Mirror
Ezhelp	17 Dec 2010	3.08	4.3 MB Mirror	Patches	4.7 MB Mirror
OddJob	15 Dec 2010	1.27	1.6 MB Mirror		4.8 MB Mirror
File Explorer 4	13 Dec 2010	4.13	6.8 MB Mirror	Patches	6.9 MB Mirror
File Explorer 5 (paid for upgrade)	13 Dec 2010	5.28	10.3 MB Mirror	Patches	10.7 MB Mirror
Office Inside 2	13 Dec 2010	2.81	5.3 MB Mirror	Patches	5.5 MB Mirror
Office Inside 3 (paid for upgrade)	13 Dec 2010	3.53	6.5 MB Mirror	Patches	6.6 MB Mirror
CryptoNite	10 Dec 2010	1.07	3.7 MB Mirror		3.3 MB Mirror
NetTalk Book	9 Dec 2010	1.22	3.5 MB Mirror		
Secwin 3	7 Dec 2010	3.58	16.4 MB Mirror	Patches	10.2 MB Mirror
Secwin 4 (paid for upgrade)	7 Dec 2010	4.94	21 MB Mirror	Patches	21.5 Mirror
xFiles	1 Dec 2010	2.14	1.7 MB Mirror		1.3 MB Mirror

Figure 3. Downloading the installer(s)

I downloaded the C7 installer, ran the installation again, and this time it gave me the option of manual or automatic C7 version detection (Figure 4).

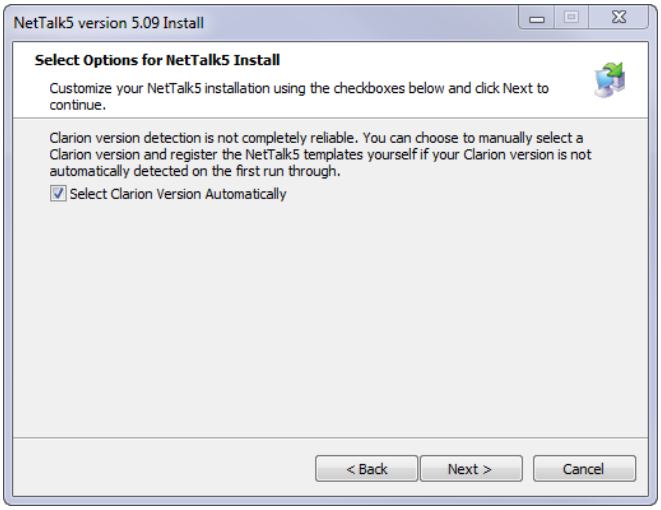

Figure 4. Running the C7 installer.

If you uncheck this option you'll see the major versions listed (Figure 5). If you leave it checked you'll see only the latest major version you have installed.

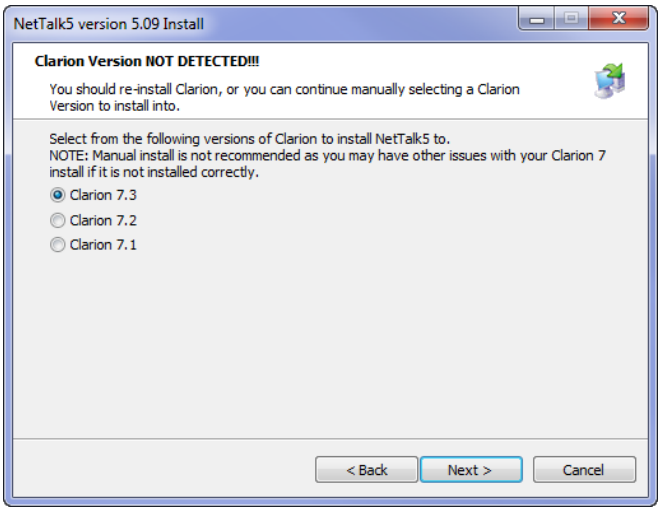

Figure 5. Choosing a Clarion version

Installer options (Figure 6) include setting up shortcuts, overriding the default folder location, and registering the templates. I accepted the defaults.

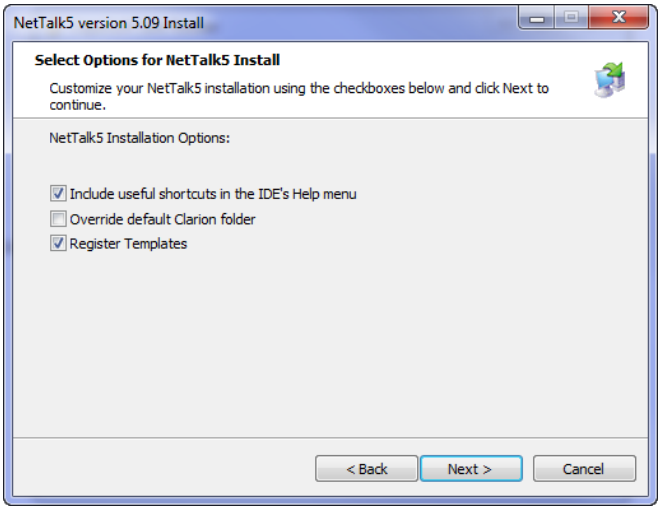

Figure 6. Installation options

Figure 7 shows the installation directory for my most recent install (I keep many C7 versions installed, each in its own directory named after the version number).

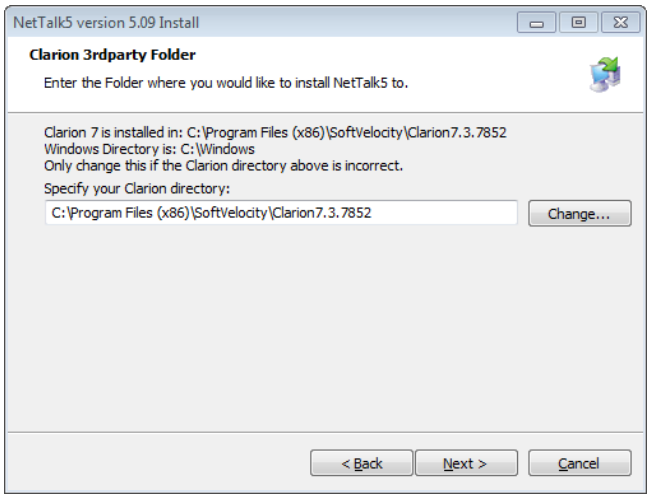

Figure 7. The install directory

The examples are installed under the C:\Users\Public\Documents\SoftVelocity\Clarion7\Accessory\Capesoft folder by default. You may wish to put them somewhere more memorable.

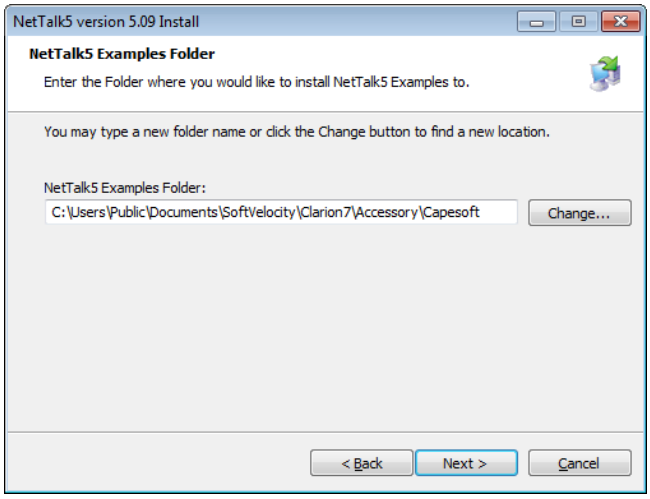

Figure 8. The install examples directory.

Once the installer is done it gives you the option of displaying the main HTML help file as well as opening the examples directory.

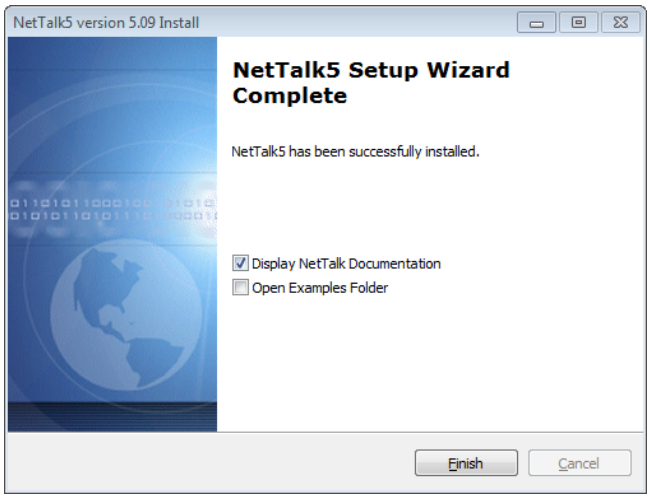

Figure 9. Installation is complete.

You can also find the Help files at{your C7 directory} \accessory\Documents\Capesoft\NetTalk. The file displayed by the installer is NetTalkIndex.htm, although this file is missing sections for the Web Client, Email, News and NetSimple. The document is marked as applying to NetTalk 5 (5.05 in my version) and according to Bruce Johnson the documentation is currently undergoing reorganization and reformatting. There is a massive amount of documentation for NetTalk so this task may take some time.

If you load up NetTalk.htm you'll get the NetTalk 4.53 docs, which do cover the missing topics in the 5.x document.

Until NetTalkIndex.html is updated, I think you may find it more helpful to start with NetTalk.htm. Figure 10 shows the top portion of that page.

Figure 10. NetTalk.htm

I had some difficulty with the Javascript popup menus in Firefox 3.63; clicking on the menu items did not take me to the specified location. The menus worked fine in IE 8, however.

Pay special attention to the "Recommended Reading" sections, as these will help give you an overview of NetTalk's capabilities, which are considerable. These sections include:

- I'm New to NetTalk
- Introduction to NetTalk
- Jump Starts (great for new users and beginners)
- Web Server FAQ (good starting place for Web Server users)
- Examples

and

- What is a client and a server?

- Using NetTalk in your Application

NetTalk ships with three kinds of classes/objects:

- NetAuto objects
- NetSimple objects
- NetDUN objects

NetAuto objects use Capesoft's NetAuto protocol, which sits on top of TCP/IP. Because this is CapeSoft's own protocol you can't use it to communicate with non-CapeSoft servers, but if you control both ends of the communication channel then NetAuto is a very easy way to establish communication between programs, primarily across a LAN).

NetSimple objects work with existing protocols, including SMTP and POP3 for email, FTP for file transfer, NNTP for newsgroups, and HTTP/HTTPS for web sites. All of these objects are available as both clients and servers except for NNTP, which is client only. NetSimple includes the ability to create browse/form web applications that have their own web server built in.

NetDUN objects let you manage and use dial-up connections.

The examples

Examples are available for all three kinds of NetTalk objects. Figure 11 shows the directory listing.

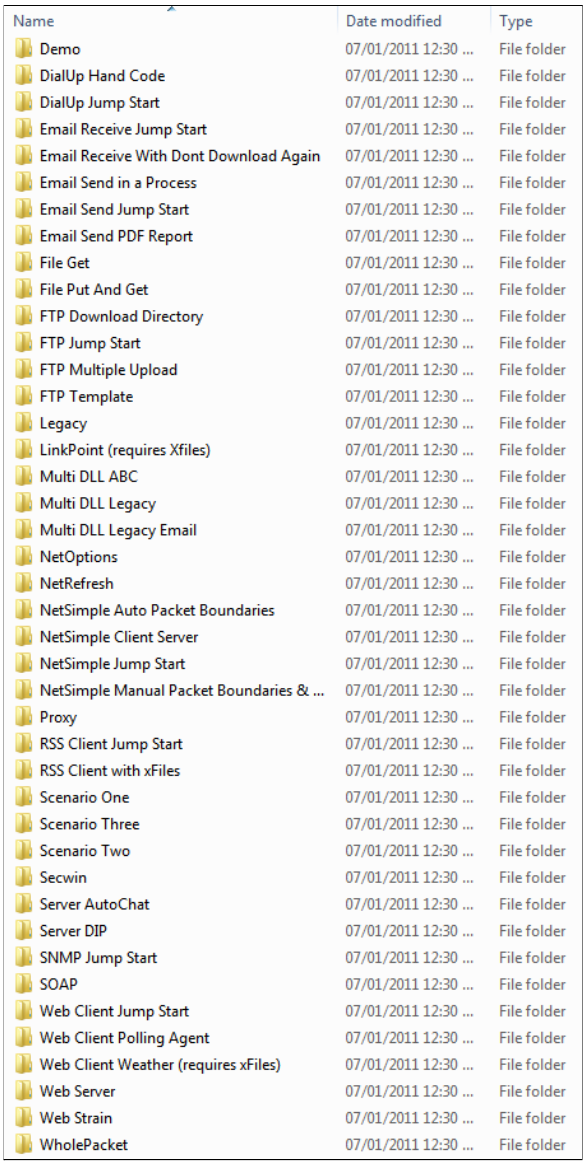

Figure 11. NetTalk examples

There are three kinds of examples. The Jump Start examples correspond to Jump Start sections in the documentation, and are intended to get you up and running on a particular feature as quickly as possible.

The three "scenario" examples cover the basics of the NetAuto protocol, including:

- Sending/receiving single packets (time client and time server)
- Broadcasting packets from one client to another client (chat application)
- Multi-packet transfer

The scenarios cover the use of the NetAuto objects, but as the docs note you may want to go through these scenarios to get a better understanding of the basics of network communication.

The remaining examples are, well, examples. The most comprehensive of these is the NetDemo application, which you can find under the examples Demo directory.

NetDemo was the first example I tried to load, and I got a generation error:

```
GEN: Registry item NetTalk NetWebHandleRequest no longer exists
```

I checked the registry and I saw that NetTalk template was not registered. As far as I know this isn't a CapeSoft problem - it's a Clarion IDE bug that sometimes prevents templates from being registered via the ClarionCL utility.

I registered NetTalk.tpl manually and was able to load and compile NetDemo.app (Figure 12).

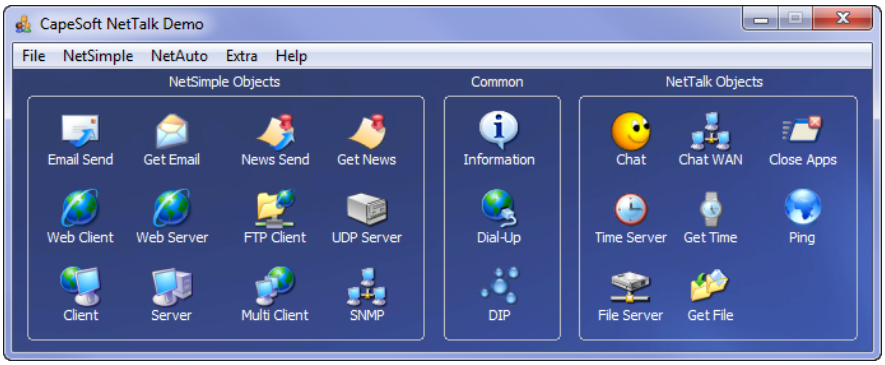

Figure 12. The NetDemo app

NetDemo's main window is organized into three parts. NetSimple demos cover the mainstream protocols such as SMTP,POP3, NNTP, HTTP, FTP, UDP and SNMP.

The Common area has a link to a help window, a dial-up networking example, and a dynamic IP (DIP) example.

The NetTalk Objects area contains NetAuto demos for chat, application control, time synchronization and file transfer.

For my first test I downloaded the list of newsgroups (Figure 13).

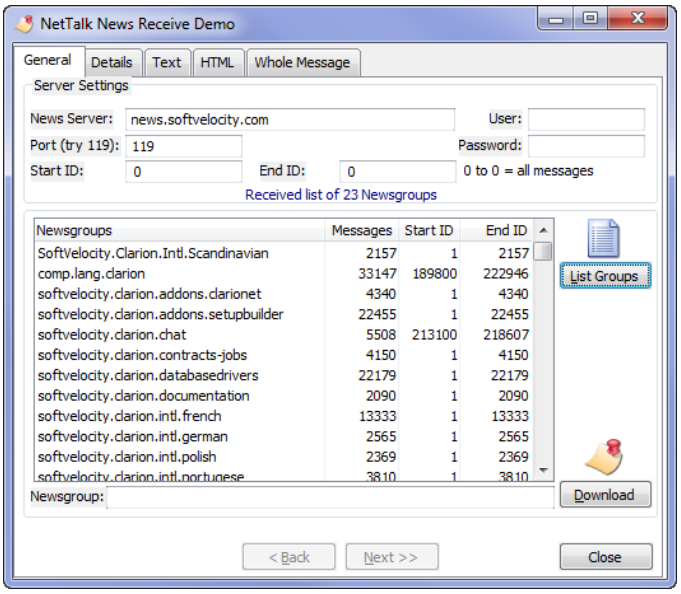

Figure 13. Downloading the newsgroup list.

I then selected comp.lang.clarion and downloaded some messages, which I viewed on the Details tab (Figure 14).

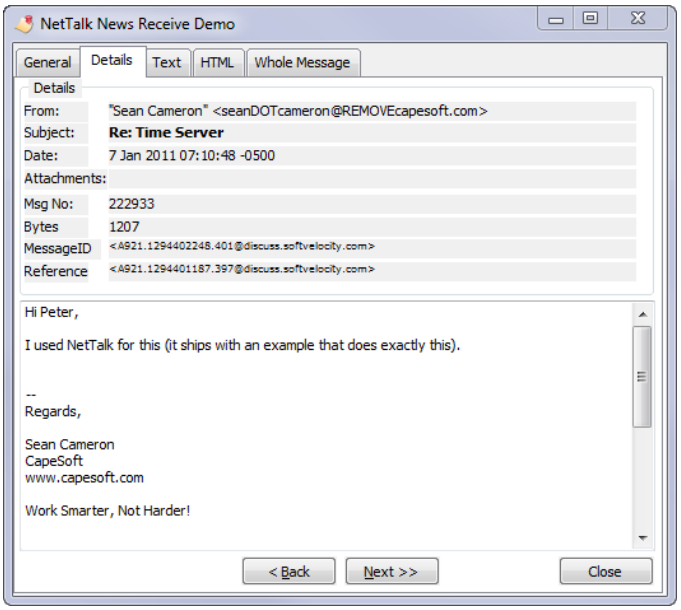

Figure 14. Viewing a newsgroup message

The only obvious problem was the lack of transparency on the entry fields, which is a known change in C7 with a template fix. I went to Global Properties, AppSettings tab and clicked on Application Manifest. I checked the Make controls inside the TAB transparent option (Figure 15). There are over 120 example applications, all of which are deliberately shipped in Clarion 5.5 format which allows them to be used in all versions of Clarion from 5.5 forward.

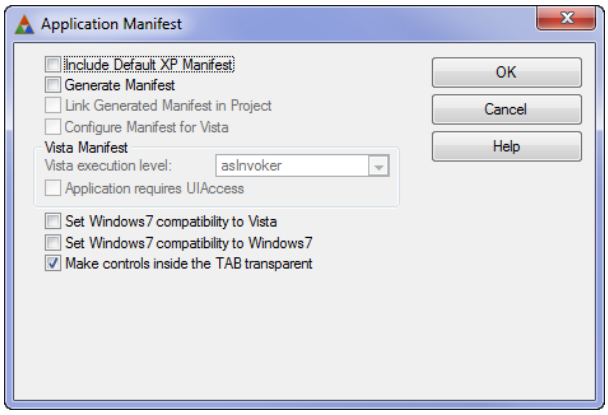

Figure 15. Cleaning up the demo

Figure 16 shows the Web Client Demo; I've retrieved the header from www.clarionshop.com (which redirected to https://www.clarionshop.com); the header shows that the server is NetTalk 4.53.

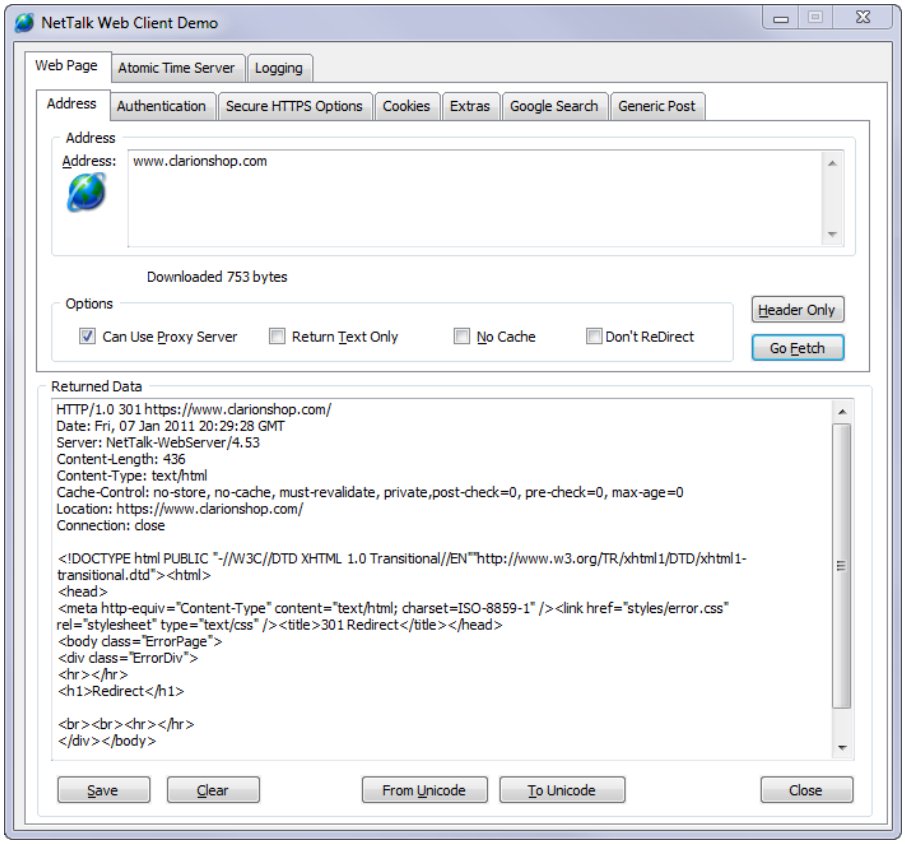

Figure 16. The Web Client demo

I also tried the Web Server example, which detected that I already had a web server running on port 80, and assigned itself to port 82.

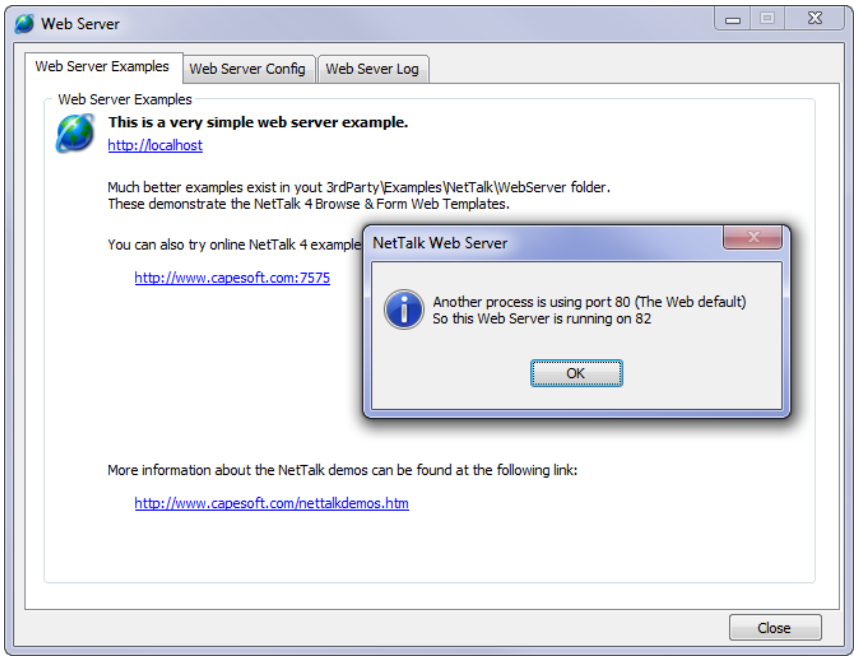

Figure 17. The Web Server example

I clicked on the hyperlink to verify the server was running, and got the page shown in Figure 18.

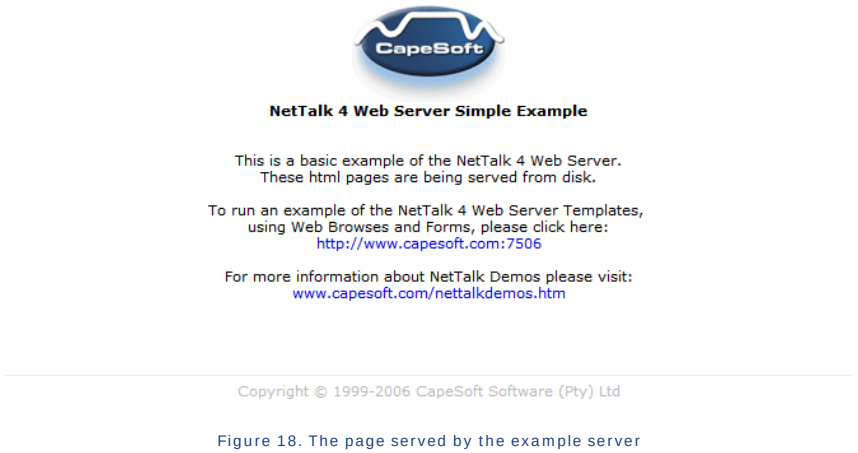

Figure 18. The page served by the example server

In Figure 19 I launched two copies of the demo app, and two instances of the chat window. (The other window looks the same except with a different name.) Interestingly if I launched two instances of the chat window from the same app no communication took place.

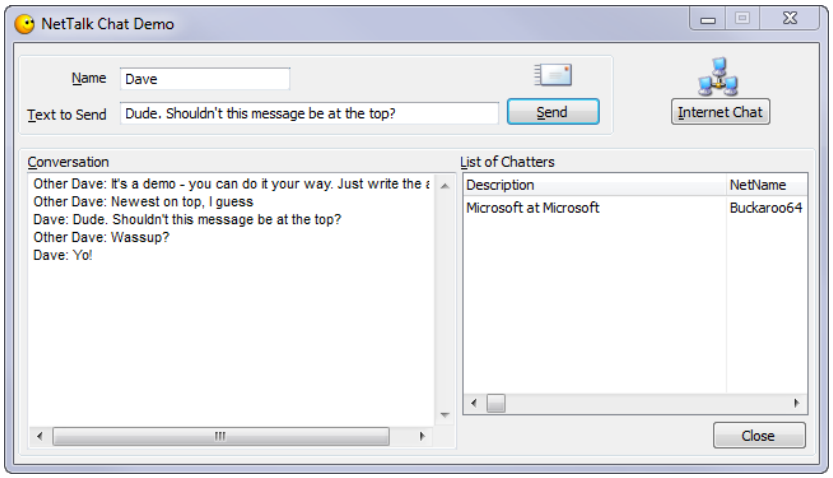

Figure 19. The Chat example

Similarly, I had to load the Time Server example from one app and the Get Time example from another app to get those two processes to communicate. As with chat, this is is how these apps would work in real life anyway.

All of the NetAuto examples are intended for LAN use. Where the option is given to try these over a WAN or the Internet you are warned that this is probably not advisable as proxy servers and some firewalls will not allow the NetAuto packets to pass through.

Templates and classes do much of the grunt work in NetTalk; in some of the examples, such as Chat, there's very little embed code.

Figure 20 shows the template list. The procedure templates are for browse/form web application (which I'll cover in Part 2). The extension templates provide access to NetTalk objects, and the control templates provide commonly-needed controls, as for chat and FTP file transfer.

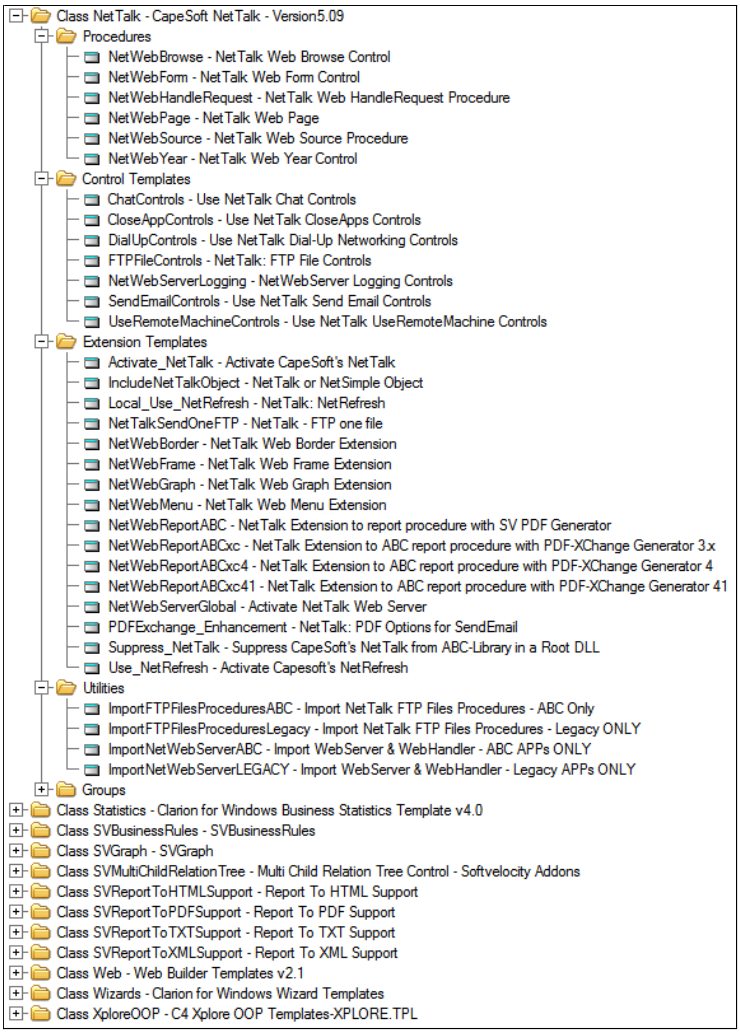

Figure 20. The template list

The tutorials

To get a feel for using NetAuto objects in an APP I went through the jumpstart for adding the ability to shutdown other instances of an application on the network, as when you need to do an upgrade and you don't want anyone using the program.

There are two components to this functionality (as there are with pretty much everything you do with NetTalk): a client and a server. We tend to think of servers as big programs that do things, and clients as smaller programs that get information from and send information to servers. But really a client is just a program sends a request and waits for a reply, and a server is just a program that listens for requests and sends replies. The client is active; the server is passive.

So if you want to enable communication between programs, you typically need both client and server functionality, and this is how the shutdown procedure works. Every instance of the application on the network is also running as a server, listening for messages. You run

the client and tell it to notify the applications of a pending shutdown. The client sends out the messages, the servers pick up the messages and perform the needed action (warning, or warning followed by shutdown).

The first thing I did was add the `Activate_NetTalk` global extension template to the app. Then, following the directions, I added the client portion of the CloseApp functionality:

- I created a new window procedure and called it from the main menu
- I populated the NetTalk `CloseAppControls` control template on the window and closed the window
- On the procedure's Extensions tab I checked that the unique application name was set. It defaults to the APP file name so I didn't change it.

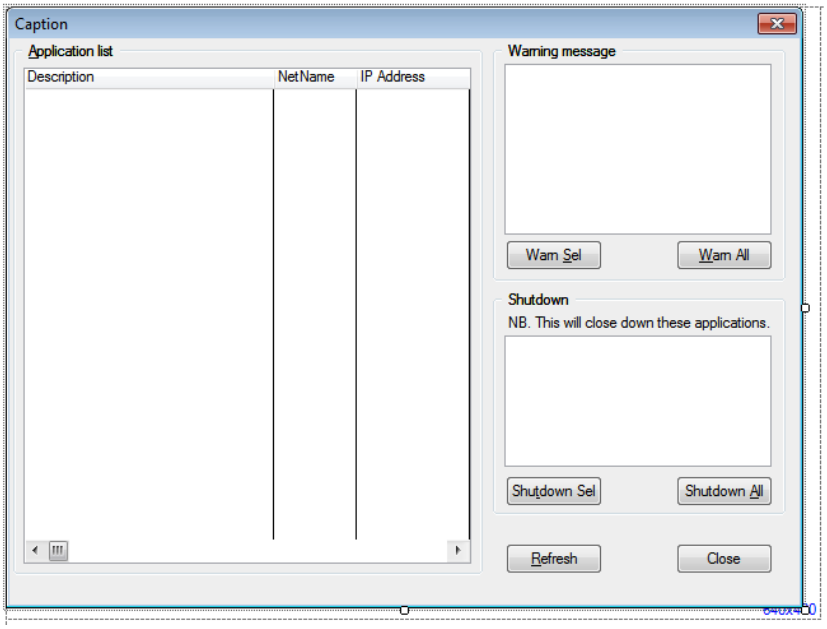

Figure 21. The populated control templates

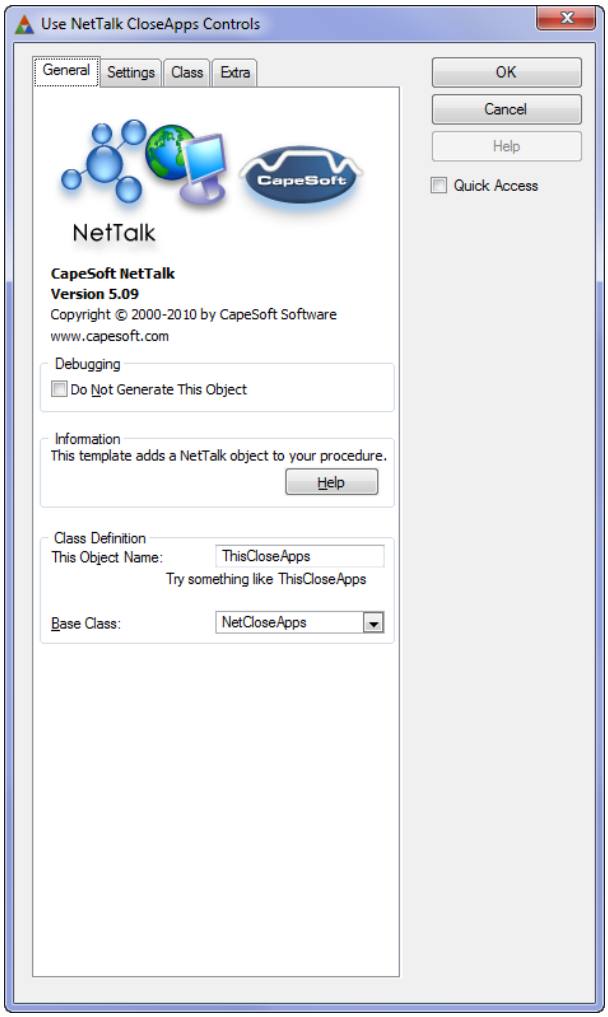

Figure 22. The General tab on the client procedure extension

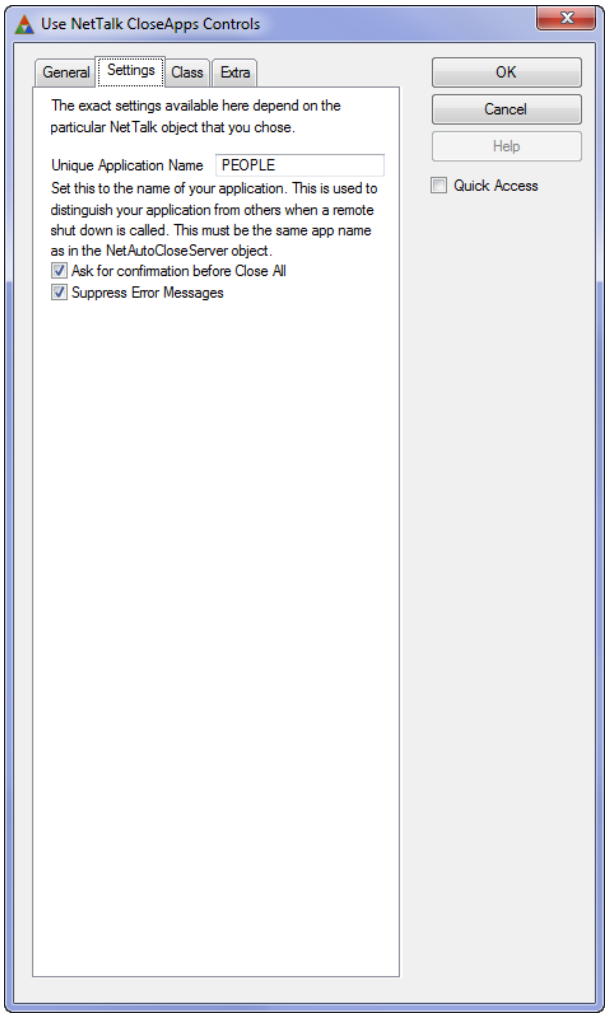

Figure 23. The Settings tab on the client procedure extension

Then I created the server portion, which was even simpler as it only involved one procedure extension template:

- On the application's frame I added the `AutoClose_Server` template
- In the This Object Name field I selected `NetAutoCloseServer`.
- I verified the Unique Application Name on the Settings tab of the extension template.

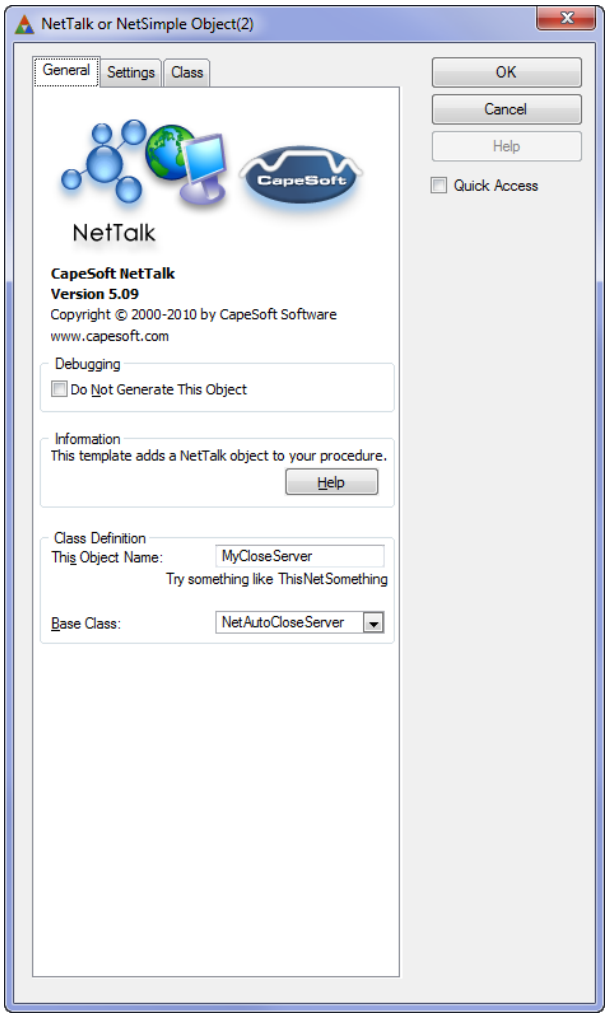

Figure 24. The General tab on the server procedure extension

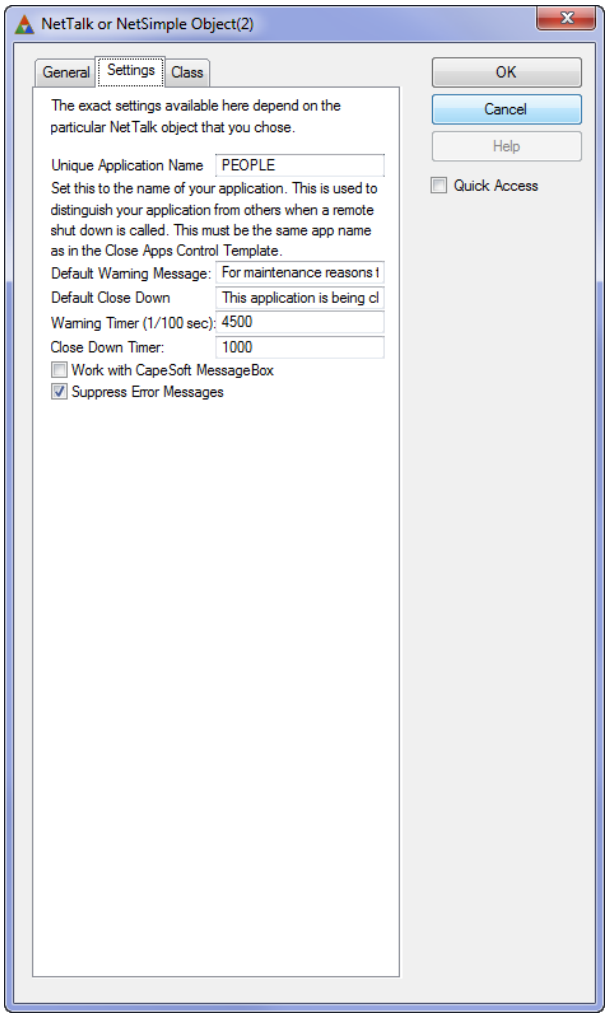

Figure 25. The Settings tab on the server procedure extension

I built the application and ran two instances. In one instance I launched my shutdown window (Figure 26).

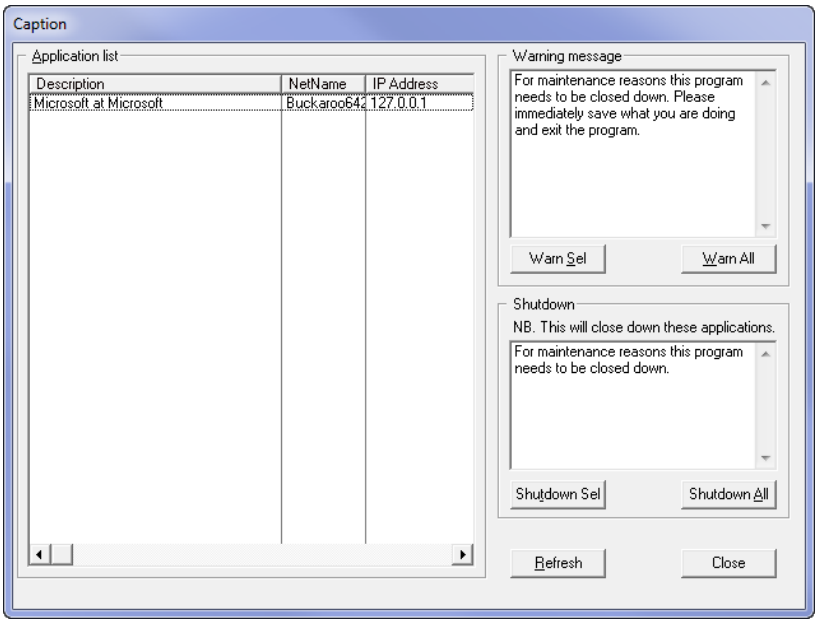

Figure 26. Shutting down another app.

The other application appeared in the list; I selected it and clicked on ShutDown Sel. The other application displayed a message box for the specified number of seconds and then closed down.

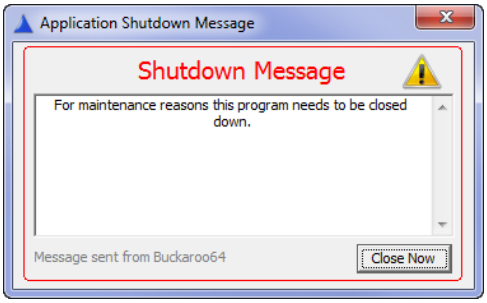

Figure 27. App being terminated

If you don't want the ability to shut down other instances to be visible to all users you have two options. One is to put the window with the shutdown controls behind some security. The other is to only put the server code in the app, and create a separate admin app with the client code. In fact, I'd probably want to get a list of programs and then choose which one to shut down.

Sending/receiving email

A lot of Clarion devs use NetTalk to send (and receive) email. The Jump Start for Sending Email section in the documentation explains how to use the `NetEmailSend` class, and walks you through the necessary calls to the class methods. Or you can just populate the SendEmailControls control template on a window. The control template will generate all of the necessary code for sending email to a server that doesn't require authentication (Figure

28).

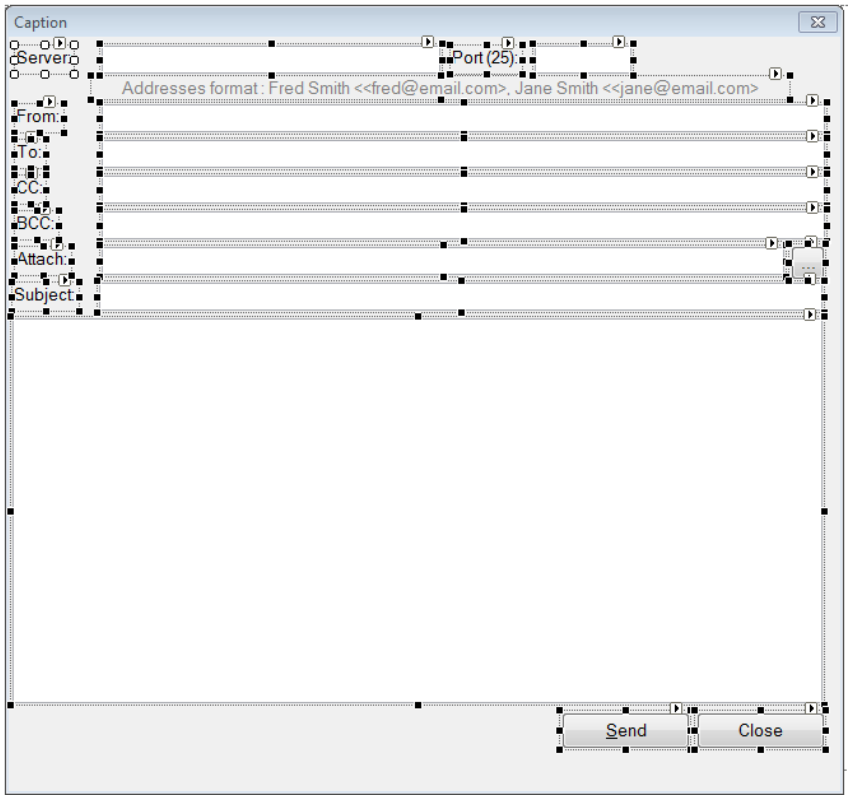

Figure 28. Populating the SendEmailControls template

My server uses authentication, so I needed to embed the following code in the Accepted embed for the Send button:

```
ThisSendEmail.AuthUser = 'authorizeduser'

ThisSendEmail.AuthPassword = 'authorizeduserpassword'
```

NetTalk sends email asynchronously, so the `SendMail` method returns immediately even though the email hasn't yet been sent. When the email has been sent, or cannot be sent, you get a notification which in the code generated by the control template is a `Message()` statement. Notifications can be handled any way you like - just derive the appropriate method in the `NetEmailSend` class and write your own code. If you don't have experience with asynchronous programming make sure you read the documentation thoroughly, as you do need to think about your code just a bit differently.

`NetEmailSend` handles multipart emails so you can use it to send attachments and HTML messages.

As NetTalk can receive email as well as send it, you might be tempted to use it to write an email server. Save your time and effort and buy the very reasonably-priced CapeSoft Email Server.

Summary

As I said at the start of this article, you can broadly divide NetTalk into two areas of functionality: browse/form web app development (which I'll cover in Part 2) and everything else.

Within the "everything else" toolset there are generally two kinds of client/server processes: those that use standard Internet protocols (which also provide the basis for browse/form development), and those that use CapeSoft's NetAuto protocol.

The NetAuto protocol (via the NetAuto objects) is useful for all sorts of communication between Clarion applications, across a local area network. This includes chat, application shutdown (and other notifications), file transfer and more.

The NetSimple objects wrap up standard Internet protocols and are useful for interacting with non-Clarion clients and servers.

I've probably touched on about one percent of what you can do with NetTalk. I'll try to cover another percent or so in Part 2. In that second part I'll look at creating browse/form applications with NetTalk, and I'll provide some concluding observations about the product as a whole.

Resources

- NetTalk home page
- NetTalk Central - a community for NetTalk users
- ClarionMag's NetTalk review from 2003

David Harms is an independent software developer and the editor and publisher of Clarion Magazine. He is also co-author with Ross Santos of Developing Clarion for Windows Applications, published by SAMS (1995), and has written or co-written several Java books. David is a member of the American Society of Journalists and Authors (ASJA).

Article comments

by Stu Andrews on January 18 2011 (comment link)

Dave, you've probably already seen it, but the Capesoft Accessories download page has had a makeover .. Go Geoff and Sean!

by Dave Harms on January 18 2011 (comment link)

Much nicer! Thanks for pointing that out, Stu.

BACK TO TOP

Clarion Tip of the Week #7: Changing Editor Font Size On The Fly

By Dave Harms

Posted January 14 2011

About fifteen years ago I started wearing glasses for work, to correct a mild astigmatism and reduce eyestrain. About five years later I broke my glasses and had to get a new pair. I had my eyes checked first, and when I first put on my new glasses I was more than a little surprised to discover the new lenses didn't just correct the astigmatism, they provided some magnification.

Five years further on, I find it difficult to read small type. That's more of a problem in print than on screen, since monitors have gotten larger and I can always change the Windows font size if I really need to.

If I just want to change the font in a given application, that's dependent on the app itself. Generally you can't change the menus and so forth - those are tied to Windows' own settings. But with text editors you can typically change the font, and of course you can do that in C7 as well.

To change the text editor font itself you have only one option: go to Tools|Options and chose the Text Editor node (Figure 1).

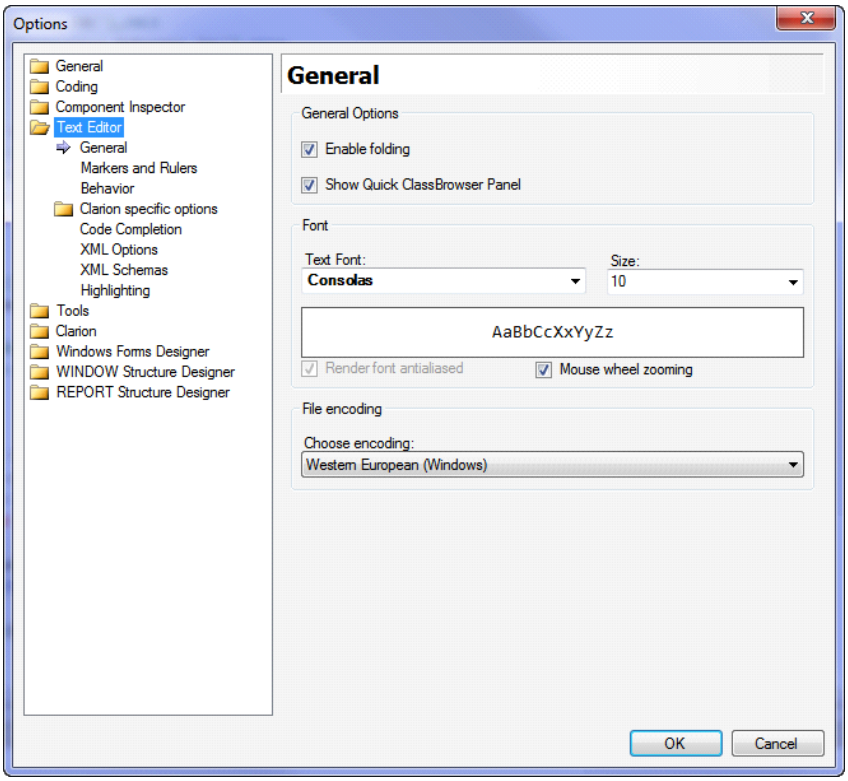

Figure 1. Editor settings

You can also change the font size here, but that's doing it the hard way. Note the "Mouse wheel zooming" checkbox. In any text editor just hold down the Ctrl key while you move the mouse wheel, and you can make the font larger or smaller.

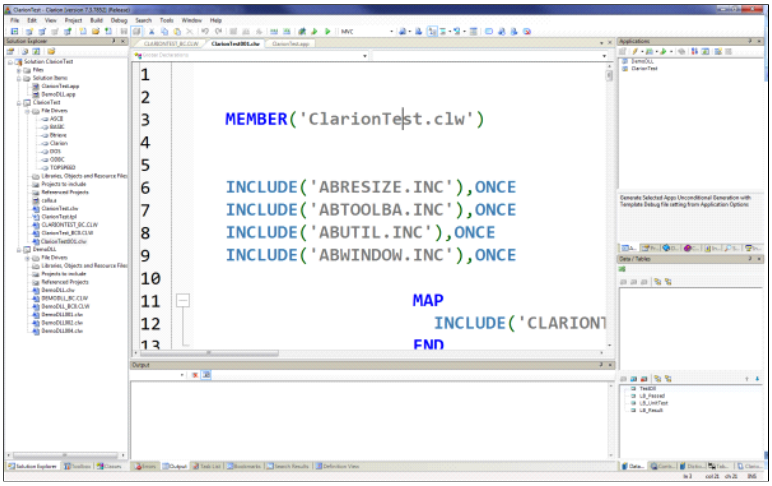

Figure 2. Using a ridiculously large font

The Ctrl-mousewheel action updates the settings in Tools|Options. If you scroll to a new text size and go to the editor options you'll see the new font size listed. And if you exit Clarion 7 and restart you'll see that the last font setting is remembered. Change the font in any text editor (full text, embeditor or embed) and that setting now applies to all other text editors

you may have open (and to any you subsequently open).

Unless you want to change the font and not the font size, just use Ctrl-mousewheel - it's far easier than going to the editor options and accomplishes exactly the same thing.

David Harms is an independent software developer and the editor and publisher of Clarion Magazine. He is also co-author with Ross Santos of Developing Clarion for Windows Applications, published by SAMS (1995), and has written or co-written several Java books. David is a member of the American Society of Journalists and Authors (ASJA).

Article comments

BACK TO TOP

MagGem: A String Class

Posted January 15 2011

Rick Martin has written some really nice articles for Clarion Magazine. One of them is about his StringClass.

If you've done any .NET or Java work you've probably come across some very capable string handling code. We don't have anything quite like that in Clarion Win32. Yes, we have a bunch of string functions, but there's something really nice about having a class for manipulating strings. It just seems cleaner, somehow.

Rick's class has the usual string capabilities, such as instring matching and extracting substrings. But it also has some nifty features like appending, replacing, substring counting, text replacing, and parsing a string into a queue. And it can handle strings of arbitrary length: you don't have to allocate memory ahead of time, you just tell the class what data it should contain, much like .NET's immutable strings.

Incidentally, the ClarionLive webinar that features this article is also the one in which I present an update on unit testing Clarion Win32 code with the ClarionTest framework. Having a framework for unit testing and test-driven development has completely changed how I do Clarion code. It can change how you code too. Check it out! (I'm also tentatively scheduled to do a followup webinar on unit testing on February 11.)

Read the article

Watch the MagGem in ClarionLive Webinar #87

Article comments

BACK TO TOP

MagGem: SQLite Is Tiny And Useful

Posted January 18 2011

It seems like Clarion devs are always on the lookout for a small, easy-to-use SQL database. The question comes up with regularity in the newsgroup.

Is SQLite that database? Probably not, but only because it's a *single-user* SQL database. Unlike most other SQL databases, there isn't a separate server program. Instead, your application loads a DLL that manipulates the database directly.

John Taylor wrote a nice little article on SQLite a while back. He pointed out all the pros and cons of using SQLite, and did some useful performance comparisons with TPS files.

He also made one comment that went right past me at the time:

> Or you may wish to take advantage of a specific SQLite feature, like high performance, or use it as a dynamic memory table without purchasing any Clarion add-on.

It's that in-memory aspect that I missed. Yes, you can use SQLite as a poor man's in-memory driver. But of late I've been thinking of yet another use for SQLite.

If you read Clarion Magazine regularly you'll know that I'm a huge fan of unit testing and test-driven development. I've created ClarionTest, a unit testing framework loosely inspired by my experiences with unit testing in .NET. And one of the things I've done in .NET is to use an SQLite in-memory database in conjunction with an NHibernate data layer. In short, this combination lets me create a database in-memory at runtime, populate it with data, and run a bunch of tests against that data.

Using an in-memory database has several advantages for testing. First, because the database is recreated each time you don't have to worry that the schema isn't up to date.

Second, because you're creating the test data from scratch each time, you don't need to worry about bad data. You know what you have.

It occurred to me that I might be able to do something similar in the Win32 world with ClarionTest and SQLite. As you can see from the webinar, I set up an SQLite ODBC data source with the special database name of `:memory:` and then I tried to use that data source with a simple Clarion application. But first I made sure I had the CREATE attribute set on

the app's files. And sure enough, SQLite created the tables in memory and I was able to add and update records.

I haven't had a need recently to do in-memory database testing, so I haven't gone any further with this approach. But it certainly looks promising.

Read John Taylor's article

Watch the SQLite MagGem in ClarionLive Webinar #88

Article comments

BACK TO TOP

MagGem: Let Your Hamster Do The Work

Posted January 21 2011

January 2011 was a rough month for users of SoftVelocity's newsgroups. The news server seemed to be overloaded; response times were terrible, and occasionally the server seemed to be down entirely.

I never noticed that, because for years now I've been following CapeSoft's lead and using the Hamster newsgroup proxy. Hamster is a bit gray around the whiskers, but it's still effective. You tell it how to get to the SoftVelocity newsgroups, and it takes care of the tedious business of polling the server for new messages. If the news server is slow, Hamster waits patiently.

You tell your news reader (I use Thunderbird) to talk to Hamster, and unless you've got some really dodgy hardware you'll find that Hamster is always there and always ready to serve. I don't even have a separate server set up for Hamster - it just runs on my development machine. And there's a nifty HamFind search utility as well, one that works better than Thunderbird's own searching. It can even search multiple archives at once, and yes, you can download at least one such archive of old Clarion newsgroup postings.

As of late January SoftVelocity has moved the newsgroups to a new server, and performance is mostly good again. But I'm sticking with Hamster, just in case.

Read more about Hamster and HamFind.

Watch the Hamster MagGem in ClarionLive Webinar #91

Article comments

by Graham Dawson on January 29 2011 (comment link)

> Hi Dave,
>
> Just wondering whether the use of Hamster could be contributing to the newsgroup problems? ie the newsgroups are slow so people start using a program that constantly tries to retrieve messages in the background and leave it running all the time and voila the newsgroups get even slower.
>
> Note I've never used Hamster so I may be completely off track here :-)

Graham

---

by Dave Harms on January 29 2011 (comment link)

Graham, since Hamster just does what your news reader does I don't think it should specifically pose a problem.

---

by Graham Dawson on January 30 2011 (comment link)

Hi Dave,

I suppose it depends how you work.

I don't have a newsreader running all the time, I log in once or twice a day, download the messages, add replies or post new messages and then log out. So I'll only be connected to the SoftVelocity server for a few minutes at a time, and perhaps for only 30 minutes in the day.

But if I understand Hamster use correctly, it will sit there periodically attaching to the server all the time the PC is running (which may be 24/7) and then you use a 'normal' reader to attach to Hamster.

So the time attached to the SoftVelocity server has got to be greater hasn't it? And if everyone starts using Hamster it will only get worse.

I suppose the ideal would be to have one site which ran Hamster, gathered all the posts to ensure they weren't 'lost' by a server crash etc, and made them available to the community. I thought that was how www.clarion-software.com worked - but it seems to have stopped, with no posts added since 18th January.

Graham

---

by Dave Harms on January 31 2011 (comment link)

Graham,

Certainly the more people who use Hamster, the more hits on the SV server. But let's say 1000 people do this (which is a whole lot - I would very much doubt more than 100). Further, let's say they all poll the server every five minutes. The actual overhead over regular use isn't the message retrieval, it's the request for the latest message number, which is a tiny amount of data and, I would think, a pretty low-cost operation for the server. That's something like 30 hits per pass. That works out to 30,000 hits every five minutes, 6000 hits every minute, 100 hits every second. I couldn't find any good stats for NNTP servers, but even a pretty wimpy web server shouldn't have much trouble serving up 100 empty pages per second.

Realistically it's unlikely to ever be more than 10% of that, IMO, and probably a lot less.

---

by Graham Dawson on February 1 2011 (comment link)

OK, thanks for the clarification.

Graham

BACK TO TOP

Tip of the Week #8: Side By Side Windows

By Dave Harms

Posted January 27 2011

When I'm working on more than app at once in C7 I have a couple of options. I can open multiple instances of C7. Or, if the apps are in the same project, I can open multiple apps in one instance of the IDE and switch between them by clicking on the appropriate tab.

But it's also possible to have multiple apps open side by side.

In Figure 1 I've just clicked on the updates.app tab and I've started to drag the tab. The tab itself doesn't move, but a drop target does appear in the middle of the window and the cursor at first changes to a no-drop cursor.

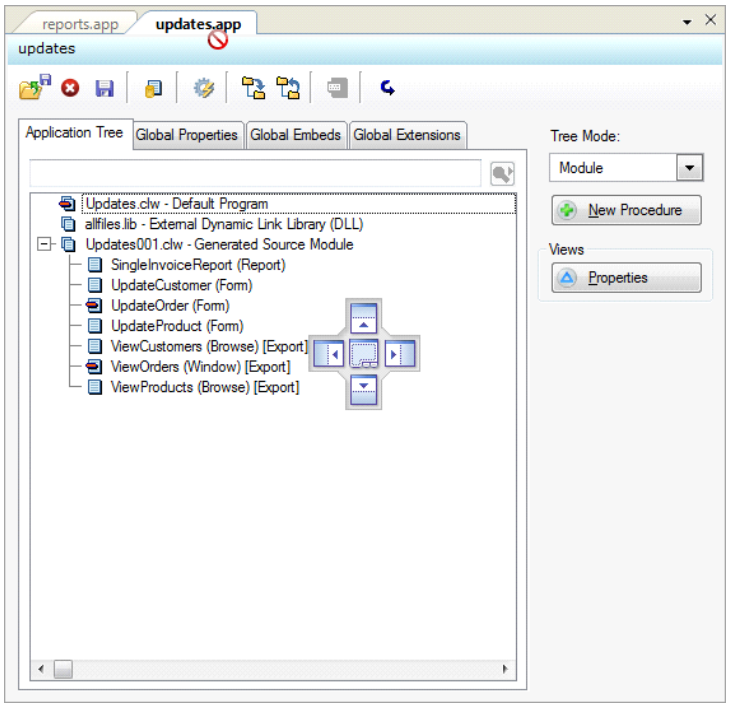

Figure 1. Beginning to drag a window tab

In Figure 2 I've dragged the tab (or attempted to - as I said the tab itself doesn't move) to the right-side drop target. As soon as I move over that target I get a highlighted region on the

screen showing where my tab will appear. When I release the mouse I'll have the two apps on side-by-side windows.

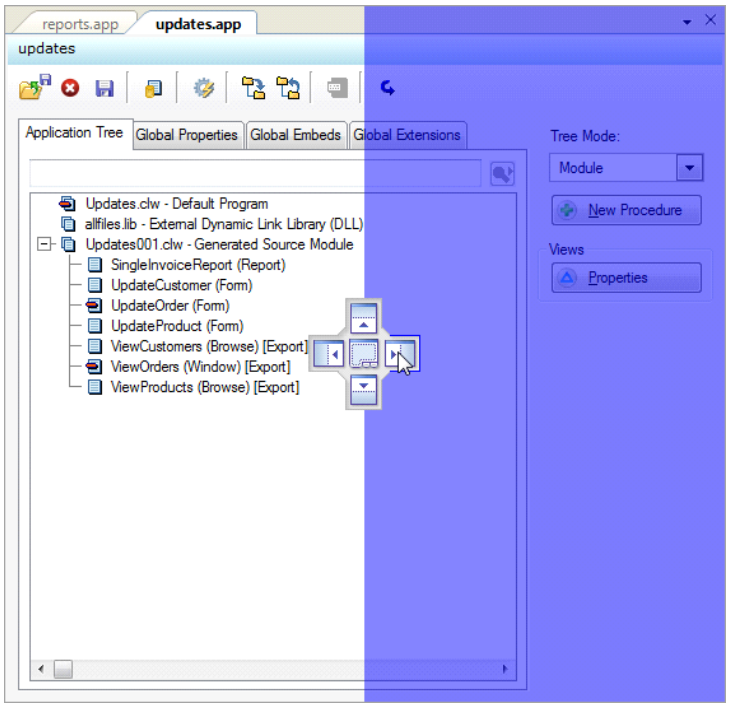

Figure 2. "Dropping" the tab

In Figures 1 and 2 I used a very narrow window so I could show all of the detail at full resolution. But you won't work with such tiny windows. Figure 3 shows a more typical setup.

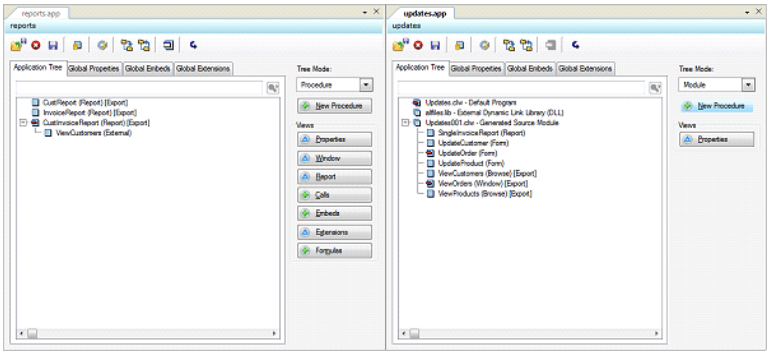

Figure 3. Two apps open side by side

In Figure 4 I've opened a report in one app and a window in another. Try that in C6!

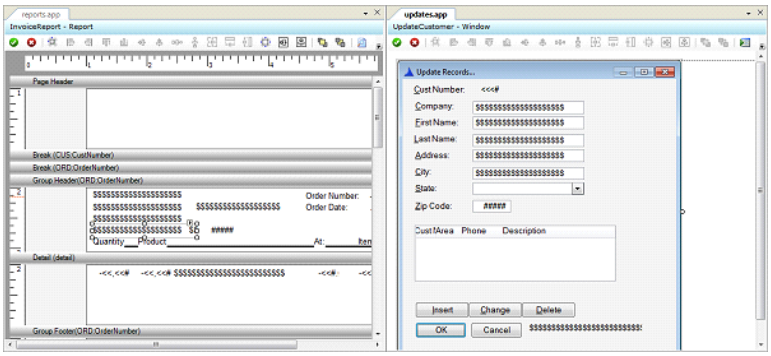

Figure 4. Editing two apps at once

Any time you have windows open on tabs you can split those windows. In Figure 5 I've opened four apps, a source file and a dictionary (if the dictionary is used by any of the open apps it will be marked as read only).

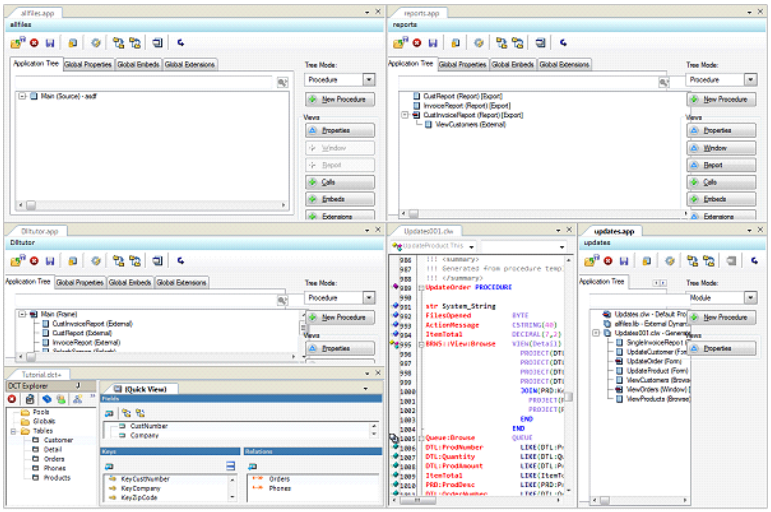

Figure 5. Getting busy

You'll also notice in Figure 5 that APP properties don't always display properly. I've also managed to crash C7 by opening a lot of windows and repeatedly resizing. So this isn't without risk, and you might want to avoid having too many windows open at once. But when it works, having multiple windows visible can be a great productivity enhancer.

David Harms is an independent software developer and the editor and publisher of Clarion Magazine. He is also co-author with Ross Santos of Developing Clarion for Windows Applications, published by SAMS (1995), and has written or co-written several Java books. David is a member of the American Society of Journalists and Authors (ASJA).

Article comments

by Lee White on January 29 2011 (comment link)

Figure 5. Getting busy

Guess I'll have to buy a bigger monitor! I wasn't aware you could open APP's side by side... good article, thanks.

by Dave Harms on January 31 2011 (comment link)

Lee, you need one of these! <g>

BACK TO TOP

MagGem: Reflection in Clarion

Posted January 27 2011

There are some mag articles on the Clarion language that are just flat-out classics. Articles by Carl Barnes and Jim Kane come readily to mind. Another is Gus Creces' excellent article on Who(), What() and Where().

If you've done some .NET programming you'll likely have come across reflection, which is the ability of a program to inspect its own code and data at runtime. It turns out that this is a very useful ability to have, because sometimes you need to deal with stuff without having any prior knowledge of what that stuff is. You might want to examine data structures, or find out what methods an object has, and so forth.

`Who()`, `What()` and `Where()` provide some reflection-like capabilities in Clarion Win32, at least on the data side. You can't use these to discover and call methods, but you can use them to examine data structures at runtime and work with that data. `Who()` returns the name of a field in a `Record`, `Class`, `Group` or `Queue`. `What()` returns a reference to the specified field, and `Where()` returns the specified field's position in the `Record`, `Class`, `Group` or `Queue`.

As Gus shows, you can get a lot of useful work done with these three functions.

Read Gus's article

Watch the MagGem in ClarionLive Webinar #93

Article comments

BACK TO TOP

Creating an SQL Query Class and Template, Part 1

By Phil Will

Posted January 28 2011

Quite often developers create a complex feature on a procedure that they would like to replicate in other procedures and applications. This is a good opportunity for a template and class. Abstracting something into a class is not always simple, but it should result in something that is easily applied and easily maintained. Among the many advantages of using classes is the ability to "encapsulate" functions (methods) and variables (properties) within a single entity, to override functions where needed, and expand functionality into new classes using previously develop classes as a starting point (inheritance).

When Bob Huff showed his end-user SQLQuery Control (Figure 1) in a recent ClarionLive! Webinar, I thought that it would make an interesting "how to convert code to a class and template" example. It even might be a useful tool for others; I asked if he would be willing to share it and happily he said yes.

This article is broken into three parts.

Part I looks at creating a simple spec and then a shell template and class. This first step does not require any functionality other than creating interlinked class and template files and instantiating the class object. The shell will include class methods for translation, debugging, and implement the WindowComponent of the WindowManager class. It will also include code methods and actual source code from the procedure, but commented out so that it will be possible to compile and develop in small steps.

Part II addresses a way to handle many controls and their associated events without much labor.

Part III fleshes out the functional part of the class code, taking advantage of Clarion's robust collection of properties and SV's ABC classes and class properties. These streamline the abstraction of procedure-specific code into generic class code.

In this case, I have created one class that handles everything. Down the road it would be worth looking at the class design "single responsibility principle", which states that a class should only do have one responsibility. A bunch of things are happening in this class that could be extracted into other classes which would make the even code more reusable.

Example query procedure

The window for the procedure Bob sent is shown in Figure 1. It has the Query controls outlined in red. Although this looks complex from an end user standpoint, he indicated that his users had learned how to use it and found it to be an effective tool.

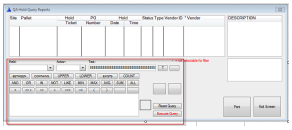

Figure 1. Original browse procedure with an SQL filter.

Initially, since I haven't done SQL programming for several years, I did not have SQL on my development computer. To get started, I converted the files in the data dictionary to TopSpeed and then ran Bob's program after commenting out those things that wouldn't compile. From running this and looking at the code, I found the following functionality:

Field. The Field drop list contains a list of fields from the files in the view. The `FROM` attribute is built by appending each field to a string. With each new selection, the field is appended to the large `TEXT` field (`QueryBuilder`). I will be referring to the field throughout the article, so keep it in mind.

Action. The Action drop list contains operators: =, <>, <=, >=, etc. The `FROM` attribute is assigned at runtime but does not contain any variables. There was no "blank" entry. Operators are appended to the `QueryBuilder` with each new selection.

Text. The Text field is a string field which can be entered by hand or used as a drop target from the Browse. When dragging/dropping, literal values are surrounded in single quotes; numeric fields are left unquoted; `DATE` and `TIME` fields are show as `DATE` and `TIME` with values in angle brackets.

T Button. The "T" button appends the contents of the `Text` field to the `QueryBuilder`.

Blank Buttons. The blank buttons perform as follows:

All. This appends the currently selected Field, Action, and Text values to the contents of the `QueryBuilder`.

Load. The next large blank button is for loading a previously saved query.

Save. The next large blank button is for saving queries in the `QueryBuilder` field.

Image. The image field referenced a green and red icon which were not included. Although this could not be duplicated with the TopSpeed files, the code showed that this would display red or green after testing the validity of each query following any change in the `QueryBuilder` field.

Reset Query. This button cleared the `QueryBuilder` and `Text` fields, cleared the browse filter, and reset the Browse.

After looking at the procedure and code, it was clear there would be some coding challenges.

Some things could be coded and test using the TopSpeed driver; when I started I didn't have an SQL database installed. SQL, with its back end language, processing, and unique requirements would obviously have to be dealt with before this could be completed.

Much of the existing coding was based on knowing the files and contents of the view including file names, field labels, and prefixes.

The class would have to deal with 36 controls including two drop lists, one `ENTRY` field, one `TEXT` field, and 32 buttons. Because many of these have similar behavior, passing a string that gets appended to the `QueryBuilder` field, I started to think about ways to simplify the code.

The specification

There were also a few behaviors I thought were worth changing. First, I felt it would be better to use drag and drop for the Query Field List, allowing you to look through the field list without forcing it to append a value to the `QueryBuilder` field. Similarly, I did not see

a need to automatically populate the Action field since this could be done using the buttons. The Action field is most useful when you drag and drop a value from the Browse to the Text field and then use the button to copy the Field, Action, and Text field to the `QueryBuilder` in one step. Finally, I wanted to avoid icons, so I chose to simply change the color of the text in the Execute Query button to red or green depending on whether the QueryBuilder contained a valid filter expression. I also wanted to disable the Save button if the filter was invalid.

There were also two things worth adding. First, the developer should have the option to not include specific files or fields in the list of fields. Secondly, I wanted to provide the missing Load and Save functionality. If a query that is being saved after one has been loaded, there should also be an option to overwrite the existing query or save it by a new name.

With all this, I had a basic specification for the Control Template and class in mind. I also generally include two other features in a class. One is a debug procedure that can be used with SysInternals. For me, this is an essential class development tool which, rightly or wrongly, has made it unnecessary to use the SV Debugger except on rare occasions. The second is to provide for internationalization. Three items should cover all the internationalization bases:

If there are any windows, there should be a `Translate(WINDOW pWin)` virtual method. If the SV Global Application Properties have "Enable Runtime Translator" turned on, the template should populate a call to SV's Translator object to translate the window.

Similarly, there should be a `TranslateString(STRING pStr),STRING` virtual method if there are any string assignments done in code.

Finally, there should be a Translation File (TRN) file with window definitions and any strings that are assigned in code so that developers can change default strings and even modify the definition of windows according to their own standards.

The template shell

The next step was to create a class template and a bare-bones, Clarion-compliant set of classes which could be used to further develop the code. Some developers I'm sure have INC, CLW, TRN, and Template shell files that they could use as a foundation. I find that approach somewhat daunting and it usually takes a number of tries before everything works.

Since I have the PD Class Generator handy, I plugged all this into the template and let the Clarion Generator do all the work. This involved creating a new application with one Window procedure to which the generator template was attached. The window controls for the control template I wanted to create were cut and pasted into the window definition and adjusted as need. The sample window is shown below.

You don't need the PD Class Generator (a commercial product which I sell) to use this example - everything is provided for you in the downloadable source, and you use those files as a starting point for your own classes and templates.

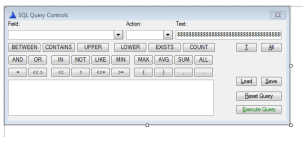

Figure 2. Window with the desired controls. These are to be added to the control template.

Earlier testing indicated that local fields with colons cause some problems, so I removed the `loc:` prefixes. I then filled out the main page of the template with all the information that would be needed to create the shell template and class definitions (Figure 3).

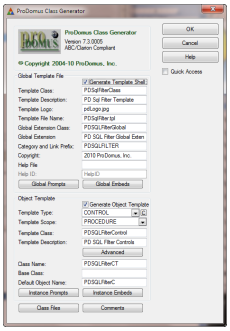

Figure 3. PD Class Generator. This is attached to the procedure with the window in Figure 1.

Identifying the "Object Template" as a CONTROL template enables the little "C" button which moves all the contols that are needed by the template into a list of controls. It also allows the identification of the ABC Browse Box Control Template as a required parent template. With this information, the Class Generator will generate a template with all the controls and their calculating their relative X and Y positions from their absolute positons on the window. Again, you don't need PD Class Generator to do all of this - you can learn how it's done by studying PDSqlFilter.tpl in the source zip.

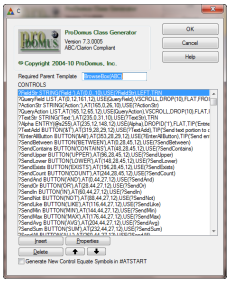

Figure 4. PD Class Generator Control Window with controls populated.

```
CONTROLS
    STRING('Field:'),AT(,,10),USE(?FieldStr),TRN,LEFT
    STRING('Action:'),AT(165,0,26,10),USE(?ActionStr)
    STRING('Text:'),AT(70,0,31,10),USE(?TextStr),TRN
```

```
    LIST,AT(-235,12,161,12),USE(QueryField),FLAT,VSCROLL,TIP('Drag from just above this field to populate a value in the query builder.
    LIST,AT(165,0,65,12),USE(QueryAction),FLAT,VSCROLL,DROP(10),FROM(' ')
    ENTRY(@s255),AT(70,0,148,12),USE(Alpha),FLAT,TIP('Entries in the list box may be dragged here.'),DROPID('1')
    BUTTON('&T'),AT(84,16,29,12),USE(?TextAdd),TIP('Send text portion to query builder')
    BUTTON('&All'),AT(34,0,29,12),USE(?EnterAllButton),TIP('Send entire line to query builder')
    BUTTON('BETWEEN'),AT(-353,0,45,12),USE(?SendBetween)
    BUTTON('CONTAINS'),AT(48,0,45,12),USE(?SendContains)
    BUTTON('UPPER'),AT(48,0,45,12),USE(?SendUpper)
    BUTTON('LOWER'),AT(52,0,45,12),USE(?SendLower)
    BUTTON('EXISTS'),AT(48,0,45,12),USE(?SendExists)
    BUTTON('COUNT'),AT(48,0,45,12),USE(?SendCount)
    BUTTON('AND'),AT(-244,16,27,12),USE(?SendAnd)
    BUTTON('OR'),AT(28,0,27,12),USE(?SendOr)
    BUTTON('IN'),AT(32,0,27,12),USE(?SendIn)
    BUTTON('NOT'),AT(28,0,27,12),USE(?SendNot)
    BUTTON('LIKE'),AT(28,0,27,12),USE(?SendLike)
    BUTTON('MIN'),AT(28,0,27,12),USE(?SendMin)
    BUTTON('MAX'),AT(32,0,27,12),USE(?SendMax)
    BUTTON('AVG'),AT(28,0,27,12),USE(?SendAvg)
    BUTTON('SUM'),AT(28,0,27,12),USE(?SendSum)
    BUTTON('ALL'),AT(28,0,27,12),USE(?SendAll)
    BUTTON('='),AT(-260,16,27,12),USE(?SendEqual)
    BUTTON('<< >'),AT(28,0,27,12),USE(?SendNotEqual)
    BUTTON('<<'),AT(32,0,27,12),USE(?SendLess)
    BUTTON('>'),AT(28,0,27,12),USE(?SendGreater)
    BUTTON('<<='),AT(28,0,27,12),USE(?SendLessEqual)
    BUTTON('>='),AT(28,0,27,12),USE(?SendGreaterEqual)
    BUTTON('('),AT(32,0,27,12),USE(?SendOpenPar)
    BUTTON(')'),AT(28,0,27,12),USE(?SendClosePar)
    BUTTON(','),AT(28,0,27,12),USE(?SendComma)
    BUTTON('.'),AT(28,0,27,12),USE(?SendPeriod)
    TEXT,AT(-260,16,288,46),USE(QueryBuilder),SKIP,BOXED,FLAT,VSCROLL
    BUTTON('&Load'),AT(319,-4,29,12),USE(?LoadQuery)
    BUTTON('&Save'),AT(34,0,29,12),USE(?SaveQuery)
    BUTTON('&Reset Query'),AT(-34,20,64,12),USE(?ClearAll)
    BUTTON('&Execute Query'),AT(0,16,64,12),USE(?ExecQuery),TIP('Execute the current query. If there is an error, it will be displayed.
  END
```

Because the control template includes local data fields, I added these fields to the Instance embeds on the template as shown below.

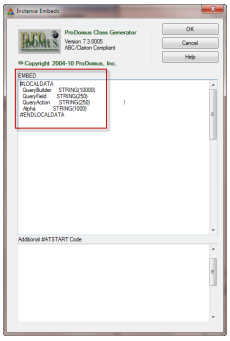

Figure 5. Instance template embed declaring data used by the controls.

The class shell

Ignoring some of the issues of making a class, I first copied all the procedure embed code into a source file and turned the mapped procedures and each of the routines into separate procedures. For the moment I did not create a `CLASS` definition or object name.

Procedure routines and local procedures

This produced a list of 16 procedures, each with some code that could be used as a starting point. All routine calls within the procedures were change from `DO Something` to `SELF.Something`, and all the calls to the procedures already defined as such where change to `SELF.ProcedureName`, i.e. `SELF.CheckQuery(querytocheck)`. Then I commented out all the code portions except except for procedure names and prototypes and return values where required in anticipation of the next step.

```
  MAP
   CheckQuery(STRING),BYTE
   DecodeQuery(STRING),STRING
   BuildAction(STRING),PROC
  END
QuerySetup          ROUTINE
GetQueryFields      ROUTINE
GetQueryActions     ROUTINE
BuildField          ROUTINE
BuildText           ROUTINE
EnterAll            ROUTINE
ClearAll            ROUTINE
ExecQuery           ROUTINE
GoGreen             ROUTINE
GoRed               ROUTINE
UndoQuery           ROUTINE
LoadQueryAction     ROUTINE
SaveQueryAction     ROUTINE
DropData            ROUTINE
AcceptQuery         ROUTINE
CheckQuery          PROCEDURE(STRING pQuery),BYTE
DecodeQuery         PROCEDURE(STRING pCodedQuery),STRING
BuildAction         PROCEDURE(STRING pAct)
```

Creating class shell methods and include files

I then plugged the procedures and their code into the PD Class Generator template – not the only way to do this but handy. I also added some procedures with no code and changed a few names to one more in line with SV naming conventions. `BuildData` thus became `TakeDropEvent`. I added `Init`, `TakeEvent`, `SetAlerts`, `Construct`, and `Destruct` methods. Some of these are shown in the list at the bottom of Figure 6.

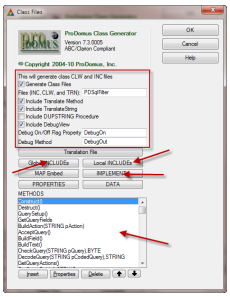

Figure 6. PD Class Generator Class Shell Setup

Event thought the procedure code was commented out, I knew that putting all that code into the methods would give me source files with much of what was need to develop the class functionality.

As shown in the red box in Figure 6, I set the name of the class source code files (`PDSqlFilter`) and clicked the check boxes to include desired translation and debug methods.

WindowComponent implementation

I knew I would need the ABWINDOW.INC in the class include file in order to utilize the `WindowComponent` interface, and the ABBROWSE.INC file in the CLW file in order to access the `BrowseClass`. In the "Implements" section I added the methods required for the `WindowComponent`.

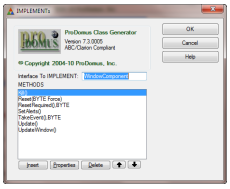

Figure 7. WindowCompant implementation setup.

In the `WindowComponent` `TakeEvent` method I added code to call the class's `TakeEvent` method and in the `SetAlerts` method I added a call to the class's `SetAlerts` method. In the Template Instance embed added code to pass my `WindowComponent` to the ABC `WindowManger` class as shown below.

```
#AT(%AfterOpeningWindow)
    SELF.AddItem(%ThisObjectName.WindowComponent)
#ENDAT
```

Using the `WindowComponent` interface this way means that the class easily then handles all events and can initialize control properties, such as Drag and Drop IDs, without the need for additional template or embed code in the procedure.

The `%ThisObjectName` symbol will contain the name of the instantiated query class object.

ABC and Clarion options

One caveat at this point is that the way this is being developed makes it applicable to ABC applications only. Use of the `WindowComponent` interface is one of the attributes that underlies this limitation. The `BrowseClass` will also make development simpler. The template generated by the PD Class Generator handles both ABC and Clarion applications so if I were serious about this being a third party product, I would do the extra work needed.

A way to handle doing both would be to create a class that worked in both ABC and Clarion and then created a derived ABC based class that added the interface and available ABC classes. The Clarion template would place the `TakeEvent` method at the start the accept loop and the `SetAlerts` method in the procedure setup area.

Summary

As a first step I created the shell template and class files with PD Class Generator - you can also do this by hand following the example of the generated code in the source zip.

These classes and templates are ready to be put in the Accessory Libsrc and Template directories respectively. The template can be registered. Although the code is not far enough along to function property, there is enough to get started. The debug and translation methods are in place.

The Control Template is set up so that controls can be populated into a test application. There is a method in place for debugging. Methods are in place for translation so that all the international readers of this article can use it when it is done. The object will instantiated if the control is populated on a window and the virtual methods will start to show up on the embed tree.

In short, there is now a framework for further development and testing . It will change a lot as development progresses, but the otherwise time-consuming boiler plate is quickly in place.

Read Part 2.

Download the source

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

Article comments

Creating an SQL Query Class and Template, Part 2

By Phil Will

Posted January 31 2011

In Part I of this article I showed a process for developing a class shell and Control Template that are ABC compliant. These included code from a procedure with a browse box and a set of query controls originally developed by Bob Huff. One of the coding challenges I identified was the large number of controls to be dealt with. In this second part I'll look at a way to include the controls and handle their events in a simple way.

In Part I had commented out all the code sections of the class libraries to be sure that they would compile without problems. I also cleared out most of Bob's dictionary, leaving enough to create a multi file view; I added a file for saved queries. This file has fields for a primary key with a single column, a query key with columns for a query category id and name, and a field for the query. This was simpler than the saved query file Bob provided, but I anticipated that the methods for identifying fields and loading and savings could be derived using embed code in the virtual methods if needed.

I then created a window procedure with a browse box and populated the query controls using the template created in Part I. I also added update buttons and an update procedure populate of some data.

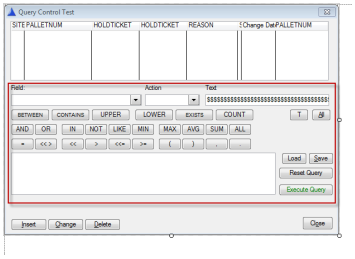

Figure 1. Test procedure with browse and control template populated.

I downloaded MS SQL 2008 R2, added the SQL Database, compiled and added some data. Since the class had no code, all of this went smoothly, happily. My strategy was to then go step by step, creating functionality in small chunks.

Handling control field equates

As pointed out in Part I, there are 36 controls, not counting the prompts. Of these 26 are buttons that send simple code strings to the query (`BETWEEN`, `CONTAINS`, etc.). One way to handle all the controls is to create properties in the class, one for each button, and code accepted events for each one. But that results in a lot of duplicated code. Instead I decided to create a `TakeAccepted` event that could handle any number of different buttons with a single block of code.

I created a queue with three fields: a control number identifier, the control field equate, and the action string if applicable. I put the queue definition in the CLW file and added a property to the class INC file.

```
FieldQT             QUEUE,TYPE
FieldNo               SIGNED
FieldFEQ              SIGNED
FieldAction           STRING(20)
                    END
!-----------------------------------------------------------------------------
PDSQLFilterCT CLASS(),TYPE,MODULE('PDSQLFILTER.CLW'),IMPLEMENTS(WindowComponent),|
LINK('PDSQLFILTER.CLW',_PDSQLFILTERLinkMode_),DLL(_PDSQLFILTERDLLMode_)
!-----------------------------------------------------------------------------
!-- Properties
FieldQ          &FieldQT
```

To create a list of field numbers, I made a list of the fields in the control template and put them into a list of itemized equates.

```
           ITEMIZE,PRE(eFld)
SendBetween         EQUATE
SendContains        EQUATE
SendUpper           EQUATE
SendLower           EQUATE
SendExists          EQUATE
SendCount           EQUATE
SendAnd             EQUATE
Etc.
```

To associate field numbers with actions, I created a group called `FieldMapG`. The first field is the number of entries, followed by pairs of `Long` and `PString` fields containing the field equate and the action, respectively. Those fields with no actions associated with them contain 'NONE' in the action field. The group fields do not need labels as they will be processed at runtime.

```
FieldMapG GROUP
  LONG(37)
  LONG(eFld:SendBetween  )
  PSTRING('BETWEEN')
```

```
  LONG(eFld:SendContains )
  PSTRING('CONTAINS')
  …
  LONG(eFld:TextAdd      )
  PSTRING('NONE')
  LONG(eFld:EnterAllButton)
  PSTRING('NONE')
         END
```

The `Construct` method creates the field queue (`FieldQ`) and the `Destruct` method disposes it. A new method, which takes the `FieldMapG` as a parameter, loops through the field pairs and adds them to the `FieldsQ` by calling a related new method called `AddField`. The `AddFields` method is called from the `Construct` method.

```
PDSQLFilterCT.AddFields PROCEDURE(*GROUP pFileMapG)
!-----------------------------------------------------------------------------
I                       SIGNED(2)
Fields                  SIGNED
ThisField               ANY
ThisNo                  SIGNED
ThisAction              STRING(20)

  CODE
    Fields= WHAT(pFileMapG,1)
    LOOP Fields TIMES
      ThisField &= WHAT(pFileMapG,I)
      ThisNo = ThisField
      ThisField &= WHAT(pFileMapG,I+1)
      ThisAction=ThisField
      IF ThisNo<>0
        SELF.AddField(ThisNo,,ThisAction)
      END
      I+=2
    END
!-----------------------------------------------------------------------------
PDSQLFilterCT.Construct PROCEDURE()
!-----------------------------------------------------------------------------
  CODE
    SELF.DebugOn=TRUE
    SELF.FieldQ &= NEW(FieldQT)
   SELF.AddFields(FieldMapG)
```

So far, the class creates the field queue and primes it using the construct method without the need for any template code. A little template code will be need to pass control equates to the `FieldQ`. The new `AddField` method is designed to take the field number and control equate to do this.

```
PDSQLFilterCT.AddField PROCEDURE(SIGNED pNo,SIGNED pFEQ=0,<STRING pAction>)
!-----------------------------------------------------------------------------
eActionParm EQUATE(4)
  CODE
   SELF.FieldQ.FieldNo=pNo
    GET(SELF.FieldQ,SELF.FieldQ.FieldNo)
    IF ERRORCODE()
      SELF.FieldQ.FieldFEQ=pFEQ
      SELF.FieldQ.FieldAction=pAction
      ADD(SELF.FieldQ,SELF.FieldQ.FieldNo)
    ELSE
      IF NOT OMITTED(eActionParm)
        SELF.FieldQ.FieldAction=pAction
      END
      IF pFEQ
        SELf.FieldQ.FieldFEQ=pFEQ !ß ADD FIELD EQUATE
      END
      PUT(SELF.FieldQ)
    END
```

Template code to add control field equates

There are two blocks of template code to handle the `AddField` calls. The first is in the `#ATSTART` embed where a multi value symbol `%FilterField` is declared along with a dependent symbol `%FilterFieldNo`. Since each of the original field names or equates is the same as what is use in the itemized list of field numbers, it is possible to use the original control names to create a list of control equates and field number equates.

```
#DECLARE(%FilterField),MULTI,UNIQUE
  #DECLARE(%FilterFieldNo,%FilterField)
  #FOR(%Control),WHERE(%ControlInstance=%ActiveTemplateInstance AND %ControlType<>'STRING' AND %ControlType<>'PROMPT')
    #ADD(%FilterField,%Control)
    #SET(%FilterFieldNo,'eFld:'&CHOOSE(SUB(%ControlOriginal,1,1)='?',SUB(%ControlOriginal,2,LEN(%ControlOriginal)-1),%Co
  #ENDFOR
```

Then in the virtual embed for the class init method, it is possible to call the addField method using these symbols in a `#FOR` loop.

```
#!-----------------------------------------------------------------------
#AT(%PDSQLFILTERMethodCodeSection,%ActiveTemplateInstance),PRIORITY(7500),WHERE(%pClassMethod='Init')
#!-----------------------------------------------------------------------
  #FOR(%FilterField)
SELF.AddField(%FilterFieldNo,%FilterField)
  #ENDFOR
```

This generates code as shown below.

```
  SELF.AddField(eFld:Alpha,?Alpha)
  …
  SELF.AddField(eFld:TextAdd,?TextAdd)
```

Event handling

At this point, the class has a queue of controls with records that identify the type of control, its field equate, and in some cases an "Action" that needs to be appended to the `QueryBuilder` field. The `WindowComponent` `TakeEvent` method will call class `TakeEvent` method for the procedures events. The next step is to code the `TakeEvent` method to handle the control events.

```
!-----------------------------------------------------------------------------
PDSQLFilterCT.TakeEvent PROCEDURE()
!-----------------------------------------------------------------------------
  CODE
    IF FIELD()
      SELF.FieldQ.FieldFEQ=FIELD()
      GET(SELF.FieldQ,SELF.FieldQ.FieldFEQ)
      IF NOT ERRORCODE()
        CASE EVENT()
        OF EVENT:Accepted
          SELF.TakeAccepted
        OF EVENT:NewSelection
          SELF.TakeNewSelection
        OF EVENT:Drop
          SELF.TakeDropEvent
        END
      END
    END
```

This code first determines if the event is a field event and then whether it is one of the fields in the `FieldQ`. If it is one of the designated fields, the code calls an appropriate method for handling the event.

```
!-----------------------------------------------------------------------------
PDSQLFilterCT.TakeAccepted PROCEDURE()
!-----------------------------------------------------------------------------
  CODE
    IF SELF.FieldQ.FieldAction='NONE' OR SELF.FieldQ.FieldAction=''
      DO TakeFieldNo
    ELSE
      SELF.BuildAction(SELF.FieldQ.FieldAction)
    END
```

The `TakeAccepted` method first checks to see if the Field Queue has an action string other thant '`NONE`' or blank. As you may recall, action items are strings that are appended to the `QueryBuilder` field such as `CONTAINS`, `BETWEEN`, `LIKE`, etc. The `BuildAction` procedure defined in the local procedure handles the appending, so this can be called directly from here. For other fields it is necessary to use the `FieldQ.FieldNo` to determine what the field is and then call methods as appropriate. Except for the reset below, these call procedures that were either declared as local procedures or routines.

Note that the `TakeEvent` method has gotten the `FieldQ.FieldNo` before `TakeAccepted` is called.

```
TakeFieldNo ROUTINE
   CASE SELF.FieldQ.FieldNo
   OF eFld:QueryBuilder
     SELF.CheckQuery()
   OF eFld:ClearAll
     SELF.Reset
     SELF.CheckQuery()
   OF eFld:ExecQuery
     SELF.ExecQuery
   OF eFld:LoadQuery
     SELF.LoadQueryaction
     DISPLAY()
   OF eFld:SaveQuery
     IF CLIP(SELF.Query)
       SELF.SaveQuery
     END
   OF eFld:QueryField
      ! Do Nothing
   OF eFld:QueryAction
      ! Do Nothing
   OF eFld:Alpha
      ! Do nothing.
   OF eFld:TextAdd
     SELF.BuildText
   OF eFld:EnterAllButton
     SELF.BuildField
     IF ~SELF.GetFieldFEQ(efld:QueryAction)
       IF CONTENTS(SELF.FieldQ.FieldFEQ)<>''
         SELF.BuildAction( CONTENTS(SELF.FieldQ.FieldFEQ) )
       END
     END
     SELF.BuildText
?  ELSE
?    ASSERT(0,'There is an unassigned field number or bad action.')
   END
```

Some field numbers have no action associated with their accepted events and accordingly "do nothing". Note that lines with question marks will not compile in release mode but will inform you of an error when debugging.

Other events

The `TakeNewSelection` method is a shell with no code. The example procedure provided by Bob Huff appended the selected `QueryField` to the `QueryBuilder`, but my specification calls for the use of drag and drop instead. The `TakeDropEvent` handles the drop from the Browse List and the `QueryField`; these will be discussed later.

Summary

At this point, the class has a way of handling the large number of controls by putting them in a queue with an identification number and a field equate. Template code has been added to associate field numbers with equates. The queue also contains several strings that can be appended to the `QueryBuilder` field when the buttons are pressed. The queue is simple, hard working, and handy. Methods are in place for handling events and calling procedures that will provide functionality to the class.

In the next and concluding article I will take the code developed in the original procedure's routines and local procedures and turn them into class code.

Download the source

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

Article comments

BACK TO TOP