Clarion Magazine

Clarion News

- » Clarion# Language Changes
- » ClarionLife News One Year Anniversary
- » PageSnip Released
- » DMC 1.1.0.5
- » Smart-Type Now Clarion 6 And Above Only
- » WhiteMarsh January Announcement
- » vuFileCount Fix Available
- » Internationalized Domain Names
- » vuFileTools 3.4
- » Valutilities Site Updated
- » SoftVelocity's New Technical Evangelist
- » iQ-XML 1.50
- » CoolFrames 1.15
- » 1stLogoDesign 50% Off Sale
- » DMC 1.1.0.4
- » DMC 1.1.0.3 Video Shows Cloning To SQL
- » DMC 1.1.0.3
- » HyperBrowze Version Support
- » Smart-Type 4.0
- » FullRecord 2.06
- » GTL 6.38
- » Compile Manager Source Code
- » SetupPROTECT SDK (sPSDK) for SetupBuilder 6 Developer
- » AFE Server Upgrade Price Increase
- » PostgreSQL Security Releases
- » PiFolio Word Reporter In Clarion.NET
- » J-Skype 2.0
- » WindowID 2.00

[More news]

- » Clarion.NET FAQ
- » Clarion# Language Comparison
- » Clarion# Language Changes Coming
- » What Can You Do With The Clarion.NET Beta?

[More Clarion & .NET]

[More Clarion 101]

Latest Free Content

- » Source Code Library 2007.12.31 Available

[More free articles]

Clarion Sites

- » DMOZ Clarion Third Party Page
- » French Clarion User Group

Latest Subscriber Content

No More Tight Loops

A tight, uninterruptable loop is fine when processing takes a very short time, but is often unacceptable for longer processes. Maarten Veenstra shows how easy it is to change your tight loops to timer-based loops.

Posted Thursday, January 31, 2008

Clarion# Language Changes Coming

Dave Harms comments on the latest Clarion# language changes, including the removal of automatic instantiation and the switch to zero-based indexes.

Posted Wednesday, January 30, 2008

What Can You Do With The Clarion.NET Beta?

No, the Clarion.NET beta doesn't have AppGen, at least not yet. So what kind of develompent can you do with the beta? Quite a lot, as David Harms explains.

Posted Friday, January 25, 2008

Create a Report using MS SQL Views

What do you do when the report you need to create requires relationships not defined in your database? You create a view, and you use the view to drive the report. Robert Johnson shows how it's done.

Posted Tuesday, January 22, 2008

Capturing Standard Output From A Console Program

When Rick Martin needed to capture the output from a version control system, his first thought of redirecting output to a text file. But there are better ways, as Rick explains.

Posted Thursday, January 17, 2008

Querying ActiveDirectory In Clarion

Microsoft's Active Directory has a myriad of uses, not least of which is managing user logins. Marty Honea shows how to use LDAP to query Active Directory from Clarion.

Posted Thursday, January 17, 2008

Lists, CHOICE, and Hidden Tabs

The interface to Steve Parker's Go To Lunch (GTL) batch compiler has undergone many revisions over the years, and the interface is getting a little crowded. The solution? Triggering hidden tabs with a list box.

Posted Friday, January 11, 2008

Using CHOOSE With PROP:SQL

Choose is a very powerful addition to the Clarion language. But what if you want to use Choose-style syntax to PROP:SQL? Steve Parker explains how it's done.

Posted Thursday, January 10, 2008

Source Code Library 2007.12.31 Available

The Clarion Magazine Source Code Library year-end release is now available. This install includes all source code from 1999-2007. Source code subscribers can download the update from the My ClarionMag page. If you're on Vista please run Lindersoft's Clarion detection patch first.

Posted Wednesday, January 09, 2008

[Last 10 articles] [Last 25 articles] [All content]

Source Code

The ClarionMag Source Code Library

Clarion Magazine is more than just a great place to learn about Clarion development techniques, it's also home to a massive collection of Clarion source code. Clarion subscribers already know this, but now we've made it easier for subscribers and non-subscribers alike to find the code they need.

The Clarion Magazine Source Library is a single point download of all article source code, complete with an article cross-reference.

More info • Subscribe now

Printed Books & E-Books

E-Books

E-books are another great way to get the information you want from Clarion Magazine. Your time is valuable; with our e-books, you spend less time hunting down the information you need. We're constantly collecting the best Clarion

Clarion Blogs

Magazine articles by top developers into themed PDFs, so you'll always have a ready reference for your favorite Clarion development topics.

Printed Books

As handy as the Clarion Magazine web site is, sometimes you just want to read articles in print. We've collected some of the best ClarionMag articles into the following print books:

- » Clarion Tips & Techniques Volume 4 - ISBN 978-0-9784034-09
- » Clarion Tips & Techniques Volume 3 - ISBN: 0-9689553-9-8
- » Clarion 6 Tips & Techniques Volume 1 - ISBN: 0-9689553-8-X
- » Clarion 5.x Tips and Techniques, Volume 1 - ISBN: 0-9689553-5-5
- » Clarion 5.x Tips and Techniques, Volume 2 - ISBN: 0-9689553-6-3
- » Clarion Databases & SQL - ISBN: 0-9689553-3-9

We also publish Russ Eggen's widely-acclaimed Programming Objects in Clarion, an introduction to OOP and ABC.

From The Publisher

About Clarion Magazine

Clarion Magazine is your premier source for news about, and in-depth articles on Clarion software development. We publish articles by many of the leading developers in the Clarion community, covering subjects from everyday programming tasks to specialized techniques you won't learn anywhere else. Whether you're just getting started with Clarion, or are a seasoned veteran, Clarion Magazine has the information *you* need.

Subscriptions

While we do publish some free content, most Clarion Magazine articles are for subscribers only. Your subscription not only gets you premium content in the form of new articles, it also includes all the back issues. Our search engine lets you do simple or complex searches on both articles and news items. Subscribers can also post questions and comments directly to articles.

Satisfaction Guaranteed

For just pennies per day you can have this wealth of Clarion development information at your fingertips. Your Clarion magazine subscription will more than pay for itself - you have my personal guarantee.

Dave Harms

ISSN

Clarion Magazine's ISSN

Clarion Magazine's International Standard Serial Number (ISSN) is 1718-9942.

About ISSN

The ISSN is the standardized international code which allows the identification of any serial publication, including electronic serials, independently of its country of publication, of its language or alphabet, of its frequency, medium, etc.

Clarion Magazine

Clarion News

Search the news archive

Clarion# Examples

Kevin Erskine has posted several Clarion# examples, including sending email and retrieving book information from Amazon.
Posted Friday, February 01, 2008

Handy Tools 12A1.01

Clarion Handy Tools has released build 12A1.01, the first of four quarterly builds planned for 2008.
Posted Friday, February 01, 2008

Gitano Releases Sterling Collection

Gitano Software has released the Sterling icon collection, containing 262 icons for a total of 14934 files. The collection includes nine sizes (16x16, 24x24, 32x32, 48x48, 64x64, 72x72, 96x96, 128x128, and 256x256) in seven formats (Windows ICO, PSD, PNG, JPG, BMP, GIF and Mac ICNS). Introductory price is $99.
Posted Friday, February 01, 2008

FullRecord 2.07 and 1.86

FullRecord 2.07 and 1.86 are now available for download. Demo available. For Clarion 6.x (Clarion 5.x is not supported in FullRecord 2). As of Clarion 7 alpha 1.5, FullRecord is full code-compatible with C7.
Posted Friday, February 01, 2008

vuSendKeys 1.1

vuSendKeys 1.1 is now available for download. Changes include: A fix to a problem with the {chr xxx} syntax; Multipliers added to macros; Five new macros; New vuActiveMyWindow function.
Posted Friday, February 01, 2008

Clarion# Language Changes

Bob Z has blogged about some important language changes coming in the next release of Clarion#, including zero based indexes for arrays and the removal of automatic instantiation.
Posted Monday, January 28, 2008

ClarionLife News One Year Anniversary

The ClarionLife news page is one year old, and has published over 1000 news items, with over 13,000 visitors.
Posted Friday, January 25, 2008

PageSnip Released

LANSRAD has released PageSnip software for Windows. PageSnip lets you print and save the web content that you want, without all the things that you don't want. You can snip an HTML article right out of the middle of a web page, then print it, create a PDF file from it, E-Mail it to a friend and even file it in PageSnip as a part of your own fully searchable knowledge base. You can even take notes about your snip and keep them with it. PageSnip lets you turn off advertising, get rid of banners and page navigation. You can gather your web content as HTML, Text or Images. PageSnip also has built-in screen capture, and you can highlight the HTML content on a web page before you print it or gather your snip. PageSnip is available via a right-click away in your web browser. PageSnip comes in both a Standard and a Professional edition. The Standard edition is designed for people who just want to snip a page or article and print it or create a PDF document from it. The Professional edition has all the features of the Standard version, and adds the ability to file the snips into a fully searchable cross referenced and indexed knowledge base so you can easily find it and use it at a later date. Prices start at $24.95 for a single-computer license of the Standard edition and $49.95 for a single-computer license of the Professional edition. There is a fully functional 30 day trial version of the Professional edition available for download. PageSnip comes with a 100% money back guarantee. Get an extra computer license for the Standard edition for only $5.00 more or the Professional edition for only $10.00 more.
Posted Friday, January 25, 2008

DMC 1.1.0.5

DMC 1.1.0.5 is now available. Changes include: Visualize any table and limit the number of records displayed; Display all the transferred data in a listbox; Support for SQL Anywhere; Option to "horizontalize" your data; Automatic mapping of date and time columns; Fixed XLS bug.
Posted Friday, January 25, 2008

Smart-Type Now Clarion 6 And Above Only

Smart-Type is a way of adding intelli-sense to your applications. Unlike other auto-complete solutions it acts on every word in a text box or entry control (or EIP) rather than just the first one. The latest version of Smart-Type is no longer compatible with C5.5 as it uses popup windows (which require C6 threading) and has a new EIP feature which derives classes from the C6 templates.
Posted Friday, January 25, 2008

WhiteMarsh January Announcement

There are five items in this January 20008 announcement: Free Metabase System; Resource Life Cycle Analysis Short Paper; Data Interoperability Community of Interest Handbook; Data Semantics Management Book; ISO/ANSI Database Standard SQL:2008.
Posted Friday, January 25, 2008

vuFileCount Fix Available

A fix to the Valutilities vuFileCount() function is now available. The bug was caused by a minor change made by the manufacturer of the underlying language.
Posted Friday, January 25, 2008

Internationalized Domain Names

Oak Par Solutions offers internationalized domain name search capabilities. You can now register .COM and .NET domain names in over 100 native languages using non-ASCII characters such as Chinese, Japanese and Arabic or in standard ASCII character format. COM Domains are on sale now for $6.95 and each domain includes free extras valued at $104.

Posted Friday, January 25, 2008

vuFileTools 3.4

vuFileTools 3.4 has been released and includes 12 new functions. The password has been changed so if you have not received your new password for vuFileTools, drop an email to sales at the domain name. The old version of vuFileTools used to install directly under Clarion. This version now installs under the 3rd Party subdirectory (where it should). Also, if you downloaded the beta version of vuFileTools 3.4b, some function calls have changed (noticeably vuSendKeys has become uVirtualkeys).

Posted Friday, January 25, 2008

Valutilities Site Updated

Valutilities has a new web site design courtesy of 1stLogoDesign.

Posted Friday, January 25, 2008

SoftVelocity's New Technical Evangelist

Clarion now has an official technology evangelist; Stu Andrews. Stu will be helping to spread the word about Clarion to the development community and providing contact and communication back to the Clarion community. He will be promoting the use of Clarion through articles, blogging, user demonstrations, and anything else he comes up with.

Posted Friday, January 18, 2008

iQ-XML 1.50

iQ-XML 1.50 handles BASE 64 encoding of objects directly in the XML. You can store and retrieve binary objects (limited to 2 meg in length at this point, but look for unlimited size in the future) inside the XML document. This is very handy to store digital signatures, large comments, documents or other larger bits of information or information that contains NULL and invalid XML data.

Posted Friday, January 18, 2008

CoolFrames 1.15

CoolFrames 1.15 beta is now available. This release fixes two MDI child window bugs.

Posted Friday, January 18, 2008

1stLogoDesign 50% Off Sale

1stLogoDesign is having a three day sale on log and design web site packages, at 50% off the regular price.

Posted Friday, January 18, 2008

DMC 1.1.0.4

DMC 1.1.0.4 is a bug fix release. In the Creation Wizard a KEY defined as NODUPE but would be created as DUP. This has been fixed.

Posted Friday, January 18, 2008

DMC 1.1.0.3 Video Shows Cloning To SQL

This short video demonstrates cloning a SQL table from a TPS table.

Posted Friday, January 18, 2008

DMC 1.1.0.3

DMC 1.1.0.3 now supports cloning to SQL. Select any source table (DAT-DBF-TPS-CSV) and create an exact cloned SQL structure. Other new features include creating a SQL table from a TPS structure in a text file, and a table creation wizard (TPS or SQL). DMC also now supports Firebird.

Posted Friday, January 18, 2008

HyperBrowze Version Support

HyperBrowze now supports Clarion 5.5, 6.0, 6.1, 6.2, 6.3, 7.0.

Posted Friday, January 18, 2008

Smart-Type 4.0

Smart-Type 4.0 has been released. Smart-Type adds "Intelli-sense" or "Auto-complete" abilities to your C5.5 and above ABC applications. It works on Entry controls, Text controls, and can be used with Edit In Place. Smart-Type is $69 during the beta program, which is nearing completion.

Posted Friday, January 18, 2008

FullRecord 2.06

FullRecord 2.06 is now available. This release has faster code generation.

Posted Friday, January 18, 2008

GTL 6.38

Steve Parker announces the availability of GTL 6.38, which features a revised user interface (please see the article in Clarion Magazine). The Generate Only option now generates .APP files but compiles .PRJ files. So, if you have multiple . PRJs which include the same .CLWs, you can generate the .APP then make each EXE/DLL.

Posted Friday, January 11, 2008

Compile Manager Source Code

Source code for Gordon Smith's Compiler Manager batch compiler is available for download.

Posted Friday, January 11, 2008

SetupPROTECT SDK (sPSDK) for SetupBuilder 6 Developer

Lindersoft has released the setupPROTECT SDK (sPSDK) for SetupBuilder 6 Developer Edition. It's free of charge for registered SetupBuilder Developer Edition users with a current subscription. The sPSDK product is not available to users of a SetupBuilder education and charity license. The setupPROTECT SDK (sPSDK) is a small, fast, and efficient software development kit specially designed for serial number and subscription key generation. The sPSDK is the "compiler" for your serial numbers or subscription keys that your development team will use to create the licenses you require. Designing a good software protection scheme will extend your software protection lifecycle, allowing you the software vendor to benefit more from increased software sales.

Posted Friday, January 11, 2008

AFE Server Upgrade Price Increase

Save $40 on the v2.5 to v2905 upgrade. This upgrade includes Clarion7 support, as it becomes available, as well as updates currently in the works for C6. v2905 introduced fax reception, improved setup, no printer driver and more.

Posted Friday, January 11, 2008

PostgreSQL Security Releases

The PostgreSQL Global Development Group has released updated versions which patch five security vulnerabilities. These releases update all current PostgreSQL versions, including 8.2, 8.1, 8.0, 7.4 and 7.3. They are considered CRITICAL and PostgreSQL DBAs and sysadmins should install the update as soon as they reasonably can. All security fixes will be included in the upcoming version 8.3 release candidate.

Posted Monday, January 07, 2008

Clarion Magazine

Clarion# Language Changes Coming

by Dave Harms

Published 2008-01-30

Bob Z has blogged about some upcoming changes to the Clarion# language, including the following:

- No more TYPE needed on class declarations - all classes are implicitly TYPEd, so all objects are by default null references and have to be instantiated
- No need to use & in class declarations, for the same reason (GROUP, FILE, QUEUE, and ANY will still require the & when declared as reference variables, however)
- Use of = rather than &= to assign objects to reference variables (but &= is still needed when working with references to value types)
- Removal of the := smart operator, which is no longer needed
- Zero-based indexes, except for files and queues.
- A new AUTODISPOSE attribute to force a DISPOSE when the object goes out of scope.

I think that last point needs a little clarification. One of the nice features of the .NET platform is automatic garbage collection. You can NEW an object, and you really don't have to DISPOSE of it; rather, the automatic garbage collection system notes when the object is no longer referenced and automatically deallocates the object's memory. The problem with automatic garbage collection is that it isn't always as quick as you might like. If you're using a limited system resource, like a font handle, you'll want to force the freeing of that resource. You can do this by calling DISPOSE. The AUTODISPOSE attribute on a declaration simply means that DISPOSE is automatically called when you exit the object's scope (e.g. leave the method in which the object is declared. This way you don't have to write the DISPOSE code yourself, and you also don't have to wait an unknown length of time before the DISPOSE actually happens.

For the most part I'm very happy about these changes, as they help bring Clarion# into line with other .NET languages; in particular the removal of TYPE and & in declarations will greatly improve the readability of Clarion# code when that code makes use of .NET libraries. While the use of the & indicator hasn't completely gone away, it's now restricted to Clarion-specific data types.

Similarly, the change in array indexing reflects compatibility with .NET while maintaining the old style for Clarion files and queues. This, however, still has the potential to cause a lot of grief, as there will inevitably be situations where 0-based code is mixed in with 1-based code. But I can live with the compromise.

There's also an interesting change to declarations and constructors. You will now be able to declare a class instance as using a particular constructor. This code:

```
s   System.String
```

creates a string reference. This code

```
s   System.String()
```

creates a string instance using the default constructor. This code

```
s   System.String('abc')
```

creates a string initialized to the value 'abc'.

Deep assignment

The deep assignment operator, which SV had earlier announced would go away, is apparently back. That makes a lot of sense to me, as .NET reflection (the ability to examine the structure of .NET objects programmatically) should make it relatively easy to implement this kind of functionality.

Interop

Finally, Z mentions some pending enhancements to make it easier to pass data between Clarion and Clarion# applications. I imagine a lot of Clarion developers will be working with both languages for some time to come, so anything that makes cross-calling easier is a good thing. At this point the trickier part is calling Clarion# from Clarion; essentially you have to create a COM wrapper around your .NET code and then use the COM object in Clarion. Calling Clarion code from Clarion# can be as simple as prototyping a DLL function in standard Clarion style (at least according to the docs - with the multitide of things to try in Clarion#, I haven't got around to that one yet).

While you're waiting for these goodies, check out Wade Hatler's series on calling .NET code from Clarion and vice versa.

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).

Reader Comments

Add a comment

Clarion Magazine

What Can You Do With The Clarion.NET Beta?

by Dave Harms

Published 2008-01-25

A lot of Clarion developers are waiting for Clarion.NET to sprout the AppGen, and rightly so. After all, while there was a time when life was simpler and quite a few Clarion developers wrote entire apps by hand, in recent years almost all of us rely to some degree on the AppGen's code generation capabilities.

So the question remains: What good is the Clarion.NET beta without the AppGen? What can you do with it now?

If you have Clarion.NET, you still have a lot of good reasons to be spending time with the beta. In this article I'll list some of the things you can do with the current release, in advance of AppGen, along with reasons why you might want to code something now or wait until later.

But before you can decide whether to tackle something in Clarion.NET you need to know what's in the box in the current release.

The compiler

Very obviously, Clarion.NET ships with the Clarion# compiler. That is, you can write Clarion# language source, and the compiler will munch that up into IL code, which can be run by the .NET platform. (It's worth noting that the IDE includes a menu option for the ILDASM command, which disassembles IL code. That means you can write some code in Clarion#, compile it, and then check the IL code to make sure it compiled as expected. This is a handy feature; among other things I've used it to determine if method attributes were being correctly compiled.)

The Clarion# compiler is quite mature; while I've found a few problems (including one related to the aforementioned method attributes), it's compiled almost all the code I've thrown at it.

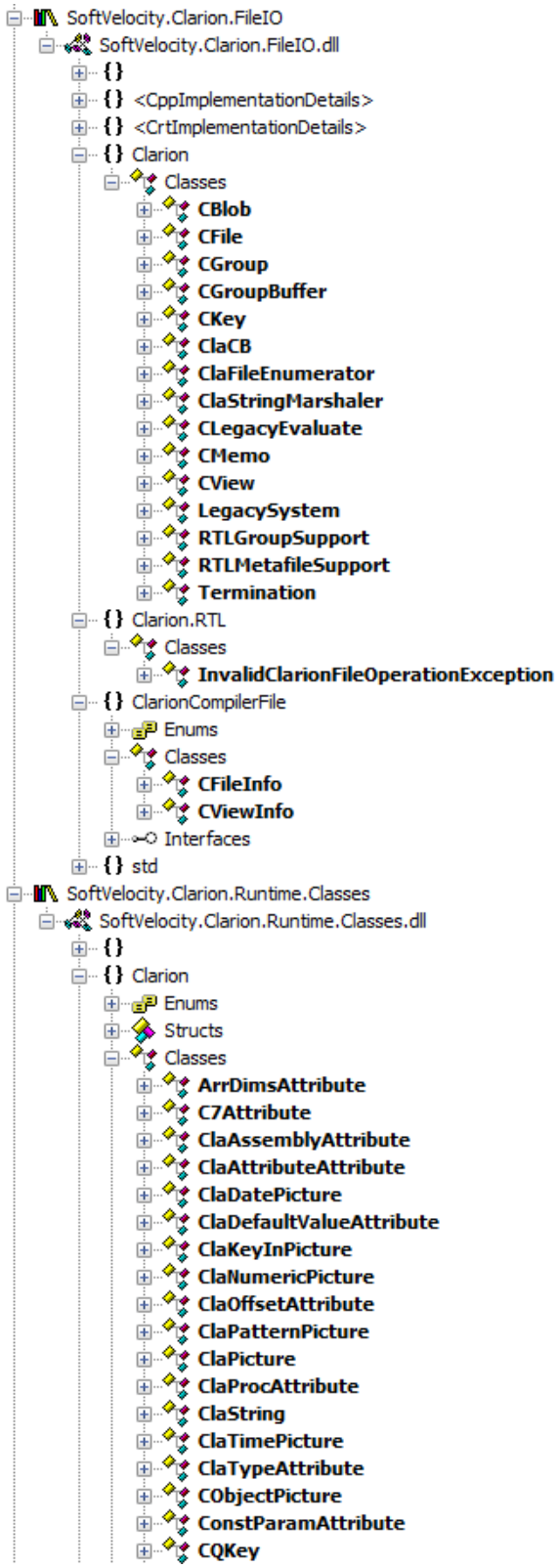
There are a few outstanding issues, however, two of which come immediately to mind. There seems to be some dispute as to whether delegates are fully implemented; as well, support for generic types is still under development.

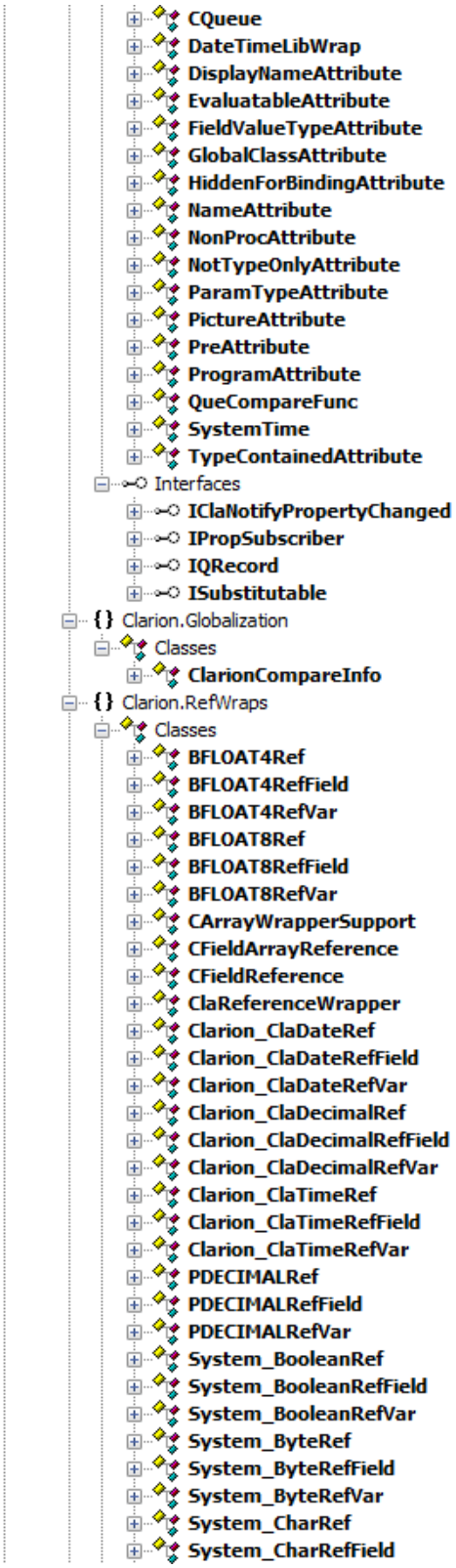
The class library

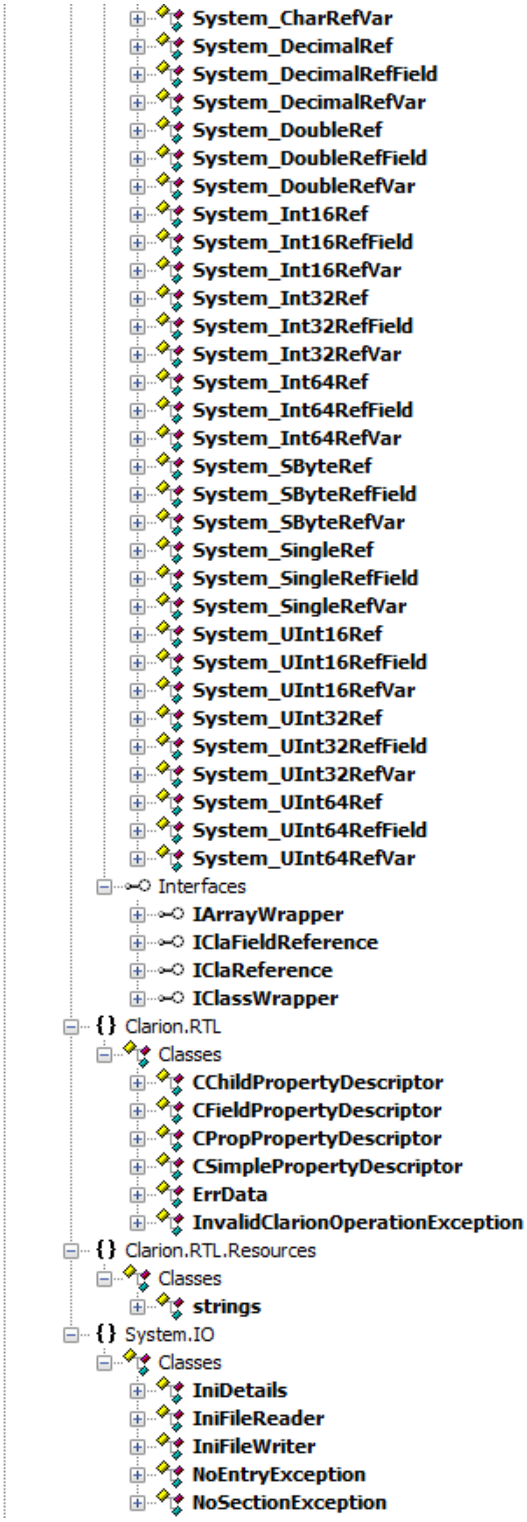
A great deal of work has been done on the Clarion.NET class library, which now appears to contain much of the functionality of the ABC class library.

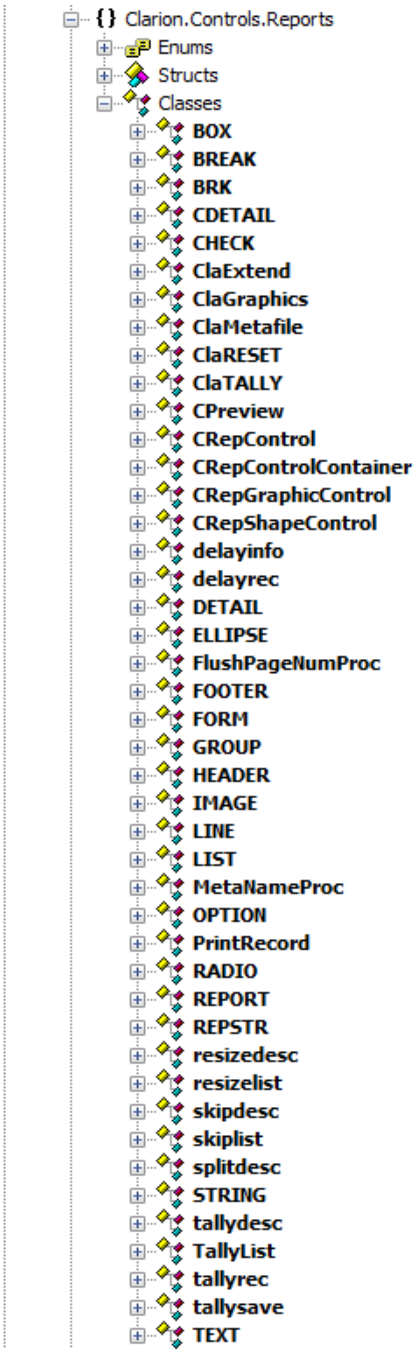
You'll recall that in Legacy (Clarion template chain) applications, the templates wrote all the code for your applications, plus your embedded code. In ABC, a lot of the functionality moved out of the templates and into the ABC class library. The templates then generated code to wire up instances of whatever ABC classes your app needed, along with the embed code in the form of virtual methods.

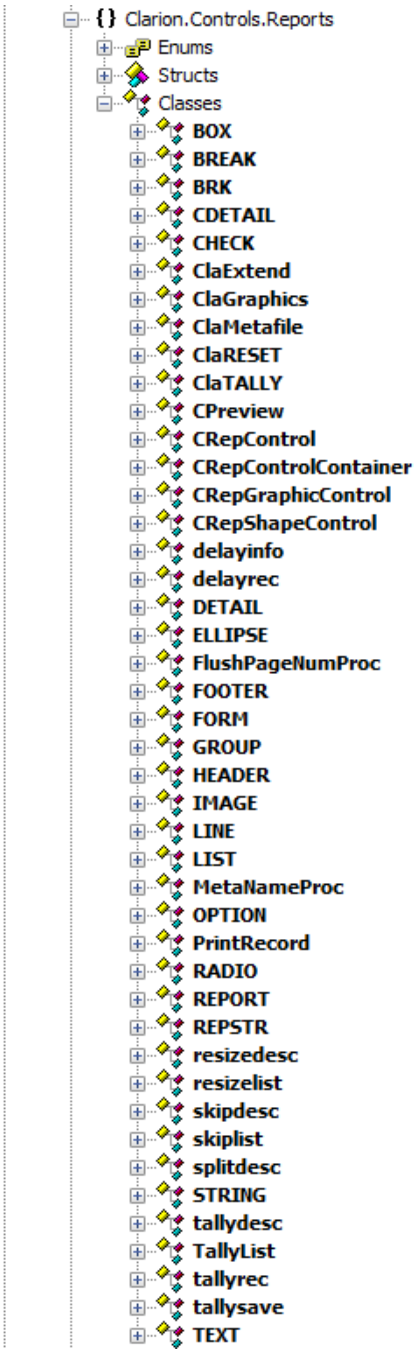
I think it's safe to assume that the .NET approach will be similar to the ABC approach, in that the templates will make use of the Clarion# class library. This library exists under the SoftVelocity root namespace. To get a look at the classes in the library I loaded up the IDE's component inspector. Figure 1 shows the runtime classes and interfaces as of build 2792.

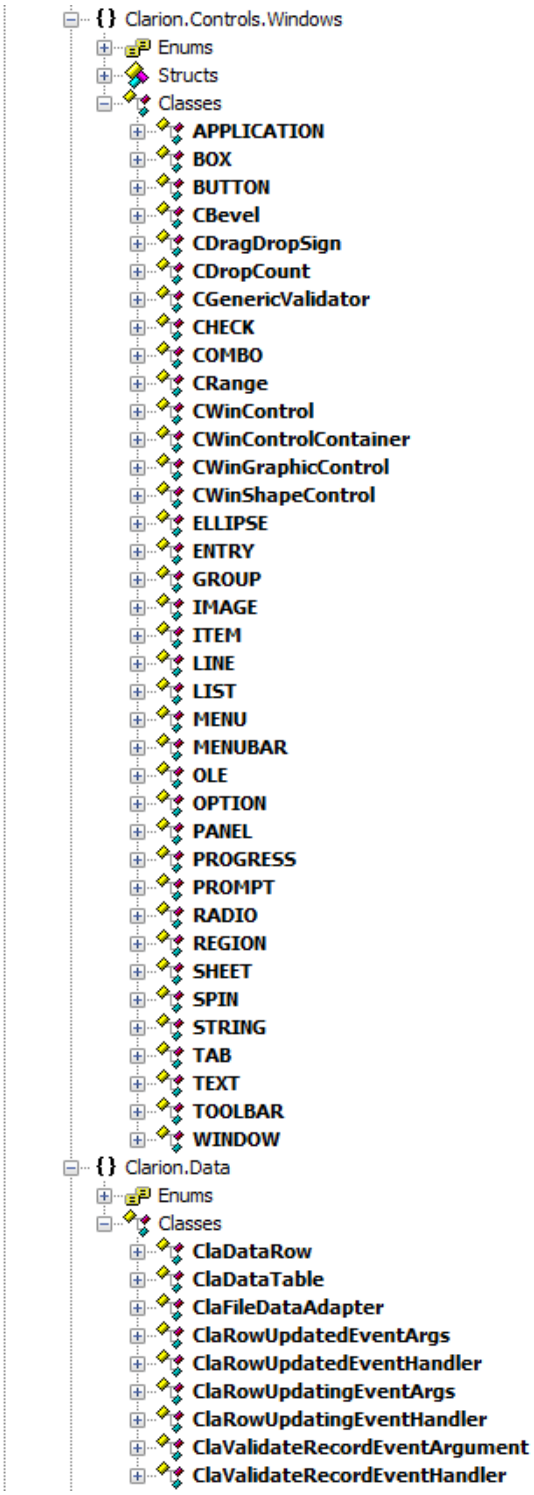

SoftVelocity.Clarion.FileIO
SoftVelocity.Clarion.FileIO.dll
{}
{} <CppImplementationDetails>
{} <CrtImplementationDetails>
{} Clarion
Classes
CBlob
CFile
CGroup
CGroupBuffer
CKey
ClaCB
ClaFileEnumerator
ClaStringMarshaler
CLegacyEvaluate
CMemo
CView
LegacySystem
RTLGroupSupport
RTLMetafileSupport
Termination
{} Clarion.RTL
Classes
InvalidClarionFileOperationException
{} ClarionCompilerFile
Enums
Classes
CFileInfo
CViewInfo
Interfaces
{} std
SoftVelocity.Clarion.Runtime.Classes
SoftVelocity.Clarion.Runtime.Classes.dll
{}
{} Clarion
Enums
Structs
Classes
ArrDimsAttribute
C7Attribute
ClaAssemblyAttribute
ClaAttributeAttribute
ClaDatePicture
ClaDefaultValueAttribute
ClaKeyInPicture
ClaNumericPicture
ClaOffsetAttribute
ClaPatternPicture
ClaPicture
ClaProcAttribute
ClaString
ClaTimePicture
ClaTypeAttribute
CObjectPicture
ConstParamAttribute
CQKey

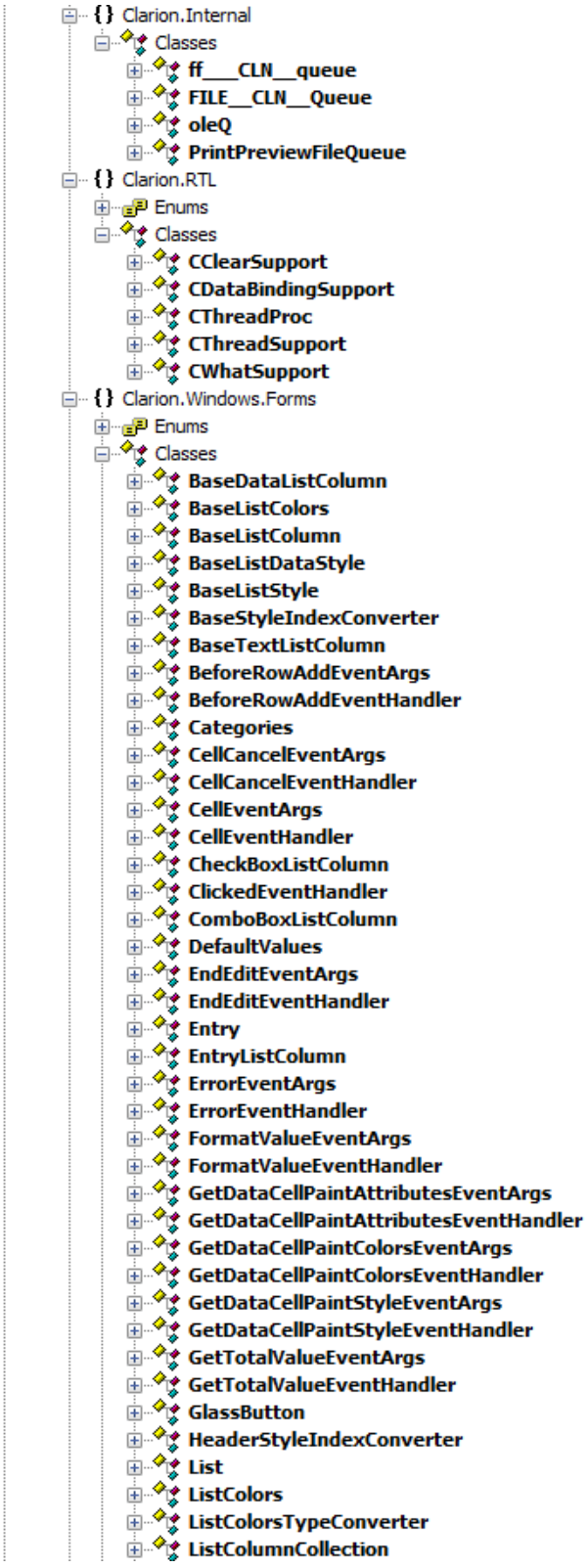

CQueue
DateTimeLibWrap
DisplayNameAttribute
EvaluatableAttribute
FieldValueTypeAttribute
GlobalClassAttribute
HiddenForBindingAttribute
NameAttribute
NonProcAttribute
NotTypeOnlyAttribute
ParamTypeAttribute
PictureAttribute
PreAttribute
ProgramAttribute
QueCompareFunc
SystemTime
TypeContainedAttribute
Interfaces
IClaNotifyPropertyChanged
IPropSubscriber
IQRecord
ISubstitutable
Clarion.Globalization
Classes
ClarionCompareInfo
Clarion.RefWraps
Classes
BFLOAT4Ref
BFLOAT4RefField
BFLOAT4RefVar
BFLOAT8Ref
BFLOAT8RefField
BFLOAT8RefVar
CArrayWrapperSupport
CFieldArrayReference
CFieldReference
ClaReferenceWrapper
Clarion_ClaDateRef
Clarion_ClaDateRefField
Clarion_ClaDateRefVar
Clarion_ClaDecimalRef
Clarion_ClaDecimalRefField
Clarion_ClaDecimalRefVar
Clarion_ClaTimeRef
Clarion_ClaTimeRefField
Clarion_ClaTimeRefVar
PDECIMALRef
PDECIMALRefField
PDECIMALRefVar
System_BooleanRef
System_BooleanRefField
System_BooleanRefVar
System_ByteRef
System_ByteRefField
System_ByteRefVar
System_CharRef
System_CharRefField


System_CharRefVar
System_DecimalRef
System_DecimalRefField
System_DecimalRefVar
System_DoubleRef
System_DoubleRefField
System_DoubleRefVar
System_Int16Ref
System_Int16RefField
System_Int16RefVar
System_Int32Ref
System_Int32RefField
System_Int32RefVar
System_Int64Ref
System_Int64RefField
System_Int64RefVar
System_SByteRef
System_SByteRefField
System_SByteRefVar
System_SingleRef
System_SingleRefField
System_SingleRefVar
System_UInt16Ref
System_UInt16RefField
System_UInt16RefVar
System_UInt32Ref
System_UInt32RefField
System_UInt32RefVar
System_UInt64Ref
System_UInt64RefField
System_UInt64RefVar
Interfaces
IArrayWrapper
IClaFieldReference
IClaReference
IClassWrapper
Clarion.RTL
Classes
CChildPropertyDescriptor
CFieldPropertyDescriptor
CPropPropertyDescriptor
CSimplePropertyDescriptor
ErrData
InvalidClarionOperationException
Clarion.RTL.Resources
Classes
strings
System.IO
Classes
IniDetails
IniFileReader
IniFileWriter
NoEntryException
NoSectionException


Clarion.Controls.Reports
Enums
Structs
Classes
BOX
BREAK
BRK
CDETAIL
CHECK
ClaExtend
ClaGraphics
ClaMetafile
ClaRESET
ClaTALLY
CPreview
CRepControl
CRepControlContainer
CRepGraphicControl
CRepShapeControl
delayinfo
delayrec
DETAIL
ELLIPSE
FlushPageNumProc
FOOTER
FORM
GROUP
HEADER
IMAGE
LINE
LIST
MetaNameProc
OPTION
PrintRecord
RADIO
REPORT
REPSTR
resizedesc
resizelist
skipdesc
skiplist
splitdesc
STRING
tallydesc
TallyList
tallyrec
tallysave
TEXT


Clarion.Controls.Reports
Enums
Structs
Classes
BOX
BREAK
BRK
CDETAIL
CHECK
ClaExtend
ClaGraphics
ClaMetafile
ClaRESET
ClaTALLY
CPreview
CRepControl
CRepControlContainer
CRepGraphicControl
CRepShapeControl
delayinfo
delayrec
DETAIL
ELLIPSE
FlushPageNumProc
FOOTER
FORM
GROUP
HEADER
IMAGE
LINE
LIST
MetaNameProc
OPTION
PrintRecord
RADIO
REPORT
REPSTR
resizedesc
resizelist
skipdesc
skiplist
splitdesc
STRING
tallydesc
TallyList
tallyrec
tallysave
TEXT


Clarion.Controls.Windows
Enums
Structs
Classes
APPLICATION
BOX
BUTTON
CBevel
CDragDropSign
CDropCount
CGenericValidator
CHECK
COMBO
CRange
CWinControl
CWinControlContainer
CWinGraphicControl
CWinShapeControl
ELLIPSE
ENTRY
GROUP
IMAGE
ITEM
LINE
LIST
MENU
MENUBAR
OLE
OPTION
PANEL
PROGRESS
PROMPT
RADIO
REGION
SHEET
SPIN
STRING
TAB
TEXT
TOOLBAR
WINDOW
Clarion.Data
Enums
Classes
ClaDataRow
ClaDataTable
ClaFileDataAdapter
ClaRowUpdatedEventArgs
ClaRowUpdatedEventHandler
ClaRowUpdatingEventArgs
ClaRowUpdatingEventHandler
ClaValidateRecordEventArgument
ClaValidateRecordEventHandler


Clarion.Internal
Classes
ff___CLN__queue
FILE__CLN__Queue
oleQ
PrintPreviewFileQueue
Clarion.RTL
Enums
Classes
CClearSupport
CDataBindingSupport
CThreadProc
CThreadSupport
CWhatSupport
Clarion.Windows.Forms
Enums
Classes
BaseDataListColumn
BaseListColors
BaseListColumn
BaseListDataStyle
BaseListStyle
BaseStyleIndexConverter
BaseTextListColumn
BeforeRowAddEventArgs
BeforeRowAddEventHandler
Categories
CellCancelEventArgs
CellCancelEventHandler
CellEventArgs
CellEventHandler
CheckBoxListColumn
ClickedEventHandler
ComboBoxListColumn
DefaultValues
EndEditEventArgs
EndEditEventHandler
Entry
EntryListColumn
ErrorEventArgs
ErrorEventHandler
FormatValueEventArgs
FormatValueEventHandler
GetDataCellPaintAttributesEventArgs
GetDataCellPaintAttributesEventHandler
GetDataCellPaintColorsEventArgs
GetDataCellPaintColorsEventHandler
GetDataCellPaintStyleEventArgs
GetDataCellPaintStyleEventHandler
GetTotalValueEventArgs
GetTotalValueEventHandler
GlassButton
HeaderStyleIndexConverter
List
ListColors
ListColorsTypeConverter
ListColumnCollection

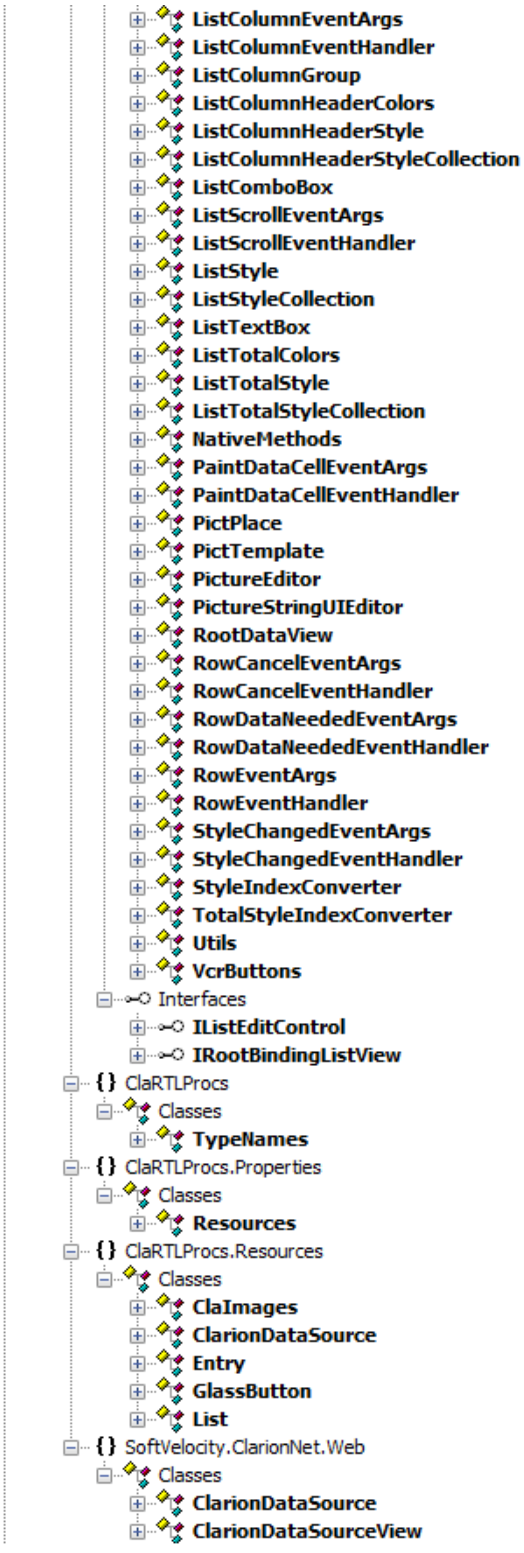


Figure 1. The Clarion# class library (view as a single image**)**

As you can see from Figure 1, there are many, many new classes, supporting various areas such as data types, file I/O, enhancements to Windows Forms, reporting and data binding.

What you don't see, for the most part, are 1:1 correspondences between ABC classes and Clarion# classes. But much of what ABC does is already in the Clarion# library. The ClaFileDataAdapter, for example is like the .NET DataAdapter, but

it lets you bind Clarion FILE structures to Window Forms list controls, for browse functionality. Other classes appear to extend Windows Forms capabilities with Clarion-like capabilities. In other words, the user interface side looks to be fairly well represented.

And don't forget about the .NET class library. Figure 1 shows SoftVelocity's enhancements and additions, but there is a vast amount of functionality already built in to .NET.

What's missing at this point is some of the database-specific code, as currently embodied in the ABC FileManager and RelationManager classes. I also don't see anything that looks obviously like locator or free-form query code.

The templates

There is a template wizard for C6 that will generate .NET desktop applications from your dictionaries, but don't expect too much of this approach. First, it's just a wizard, not a template set for APPs. It generates one-off source code, not two-way code with embeds. Second, the code generated by the wizard is in no way intended as an example of good .NET application architecture. It's more of a hybrid of Legacy and .NET code, thrown together to give Clarion.NET developers some working examples to chew on.

SV has said little about the .NET template sets, whether for desktop, mobile, or web applications. I can speculate, but I think I'll leave that for another time, except to say that I hope SV will use a template and application architecture that maximizes the ability to reuse code among desktop, web, and mobile apps. Certainly there's a lot of common ground already in the .NET framework between desktop and compact/mobile apps, and presumably Clarion# will go the same route.

So given the code embodied in the .NET class library, and the SV class library, and the current compiler, what *can* you do with the Clarion.NET beta? You can write simple desktop apps, and perhaps slightly more complete mobile and web apps.

Simple desktop apps

Randy Rogers has done more work on handcoded Clarion# desktop apps than anyone I know. It isn't easy sledding, but it can be done, although as Randy has shown the learning curve is pretty steep, partly because the documentation is pretty sparse at the moment and partly because the class library is incomplete. For instance, you'll probably find it difficult to write a full-fledged, page-loaded browse. Other tasks are considerably easier in Clarion# than in Clarion, as demonstrated by another of Randy's projects, integrating a Google Calendar into a Clarion# application.

In short, the core business for most Clarion developers, the browse/form interface to large data sets, needs better support from the class library, and would benefit a whole lot from the Dictionary Editor, AppGen, and suitable templates.

Recommendation: For desktop apps, handcoding browses and forms is a tough sell, compared with the ease of template-driven development. But there are lots of opportunities to explore Clarion# and learn some new concepts; even better, after a few decades of limited third party options, the available universe of .NET tools and accessories is going to make a lot of Clarion# developers very happy. If you want to have fun, start Googling for some cool .NET code to integrate into your apps.

Mobile apps

Clarion# is the first release of Clarion to directly support mobile application development. Mobile apps tend not to be as browse/form intensive as desktop apps, and in my opinion are a better use of Clarion# development at this time. If you're looking to add a mobile component to your application suite, this may be a good time to get started. Clarion# appears to support both PDA and SmartPhone development, but at present only PDA skins are shipping. (SmartPhones have a much more restricted control set and do not have a pointing device for user input.)

Recommendation: As with desktop development, you may want to steer away from data-intensive apps for now.

Web apps

Clarion# lets you build ASP.NET web applications, and there are basically two ways to do ASP.NET apps. One is the traditional way, employed by the vast majority of ASP.NET programmers. You create web pages which are (or can be) a mix of HTML and programming language code and your ASP.NET web server (typically IIS) compiles that code into server and client components, delivering the client bits to the user and executing the server bits as required. As far as I know, the Clarion# web templates will follow standard ASP.NET practices.

Barring a template set, you can write ASP.NET web apps the same way a lot of other developers write them: by hand.

Another way to write web apps, which is much less common than traditional ASP.NET but is gaining significant mind share, is something called MVC. MVC stands for Model/View/Controller, and is a well-established pattern for separating the user interface from the control logic from the business logic. Even Microsoft has taken note of this movement, and has released an MVC toolkit for ASP.NET 3.5.

I haven't tried Microsoft's MVC product, partly because it requires version 3.5, and partly because it's so new. But Microsoft seems to have patterned their effort after the Castle Project's Monorail, which is itself inspired by Ruby on Rails. I have been working with Monorail, and have had good success except for the aforementioned problemwith typeof in attributes, which is reportedly fixed for the next build. If testing continues to go well, I'll have much more to say about Monorail and Clarion# in future articles. Monorail uses Castle's ActiveRecord library, which is built on top of the NVelocity ORM framework.

I should also mention that Clarion.NET ships with a sample web service application, and web services are generally good candidates for hand coding, as they are essentially procedures which can be called via an Internet connection.

Recommendation: ASP.NET is the safe road, but based on my testing to date, Monorail looks like a promising alternative for rapid web development. If you plan on doing traditional ASP.NET, however, you'll probably want to wait for the templates.

Summary

For most Clarion developers, writing applications means using lots of data in a browse/form environment. Until SV ships the AppGen and some templates, that kind of development is going to remain a challenge in Clarion.NET. But you can still do a lot of useful work with the current beta, whether you're writing desktop, web, or mobile apps.

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).

Reader Comments

Add a comment

Clarion Magazine

No More Tight Loops

by Maarten Veenstra

Published 2008-01-31

Every now and then I read questions in the newsgroups about applications not responding and screens not refreshing. All the situations I've seen were caused by programmers writing tight loops and not giving the OS time to do its thing. The solution is simple: use a timer-based loop. With Clarion this is easy to write; all you need is a (hidden) window and an accept loop.

The badly-behaved loop

If you have some simple file processing, you might be tempted to write it like this:

```
Set(OneFile)
Loop
  Next(OneFile)
  If ErrorCode() Then Break .
  Two:Record = One:Record       !Some process
  Add(TwoFile)
End
```

A little quick and dirty programming job like this has its uses and will work fine when there are only a few records in OneFile. But if you have a few hundred, or perhaps a few hundred thousand records, don't be surprised that the computer won't respond or the progressbar won't update. As long as your program is stuck in the loop your application can't respond to the user.

The well-behaved loop

The way to fix a tight loop is to use a Clarion Window procedure and a timer event. Timer events fire at specified regular intervals, from a hundred times a second up to once every 65 seconds. Here's how you create a window with a timer event:

Insert a new procedure and pick the Window - Generic window handler procedure type from the Templates tab. On the properties screen, press the Window button and choose Simple Window.

In the window formatter, select the window and press Enter to call up the window properties screen. Enter a 1 (one) in the Timer field, on the Extra tab. Press Ok and Exit and press the Embeds button.

Go to the Window Managers Init property, after the Open the window embed point, and insert a Source embed. Enter two lines of code:

```
0{Prop:Hide} = True     !Hide the window
Set(OneFile)            !This is line 1) from the code above
```

Exit and save. Then go to the Procedure Routines embed point and insert a Source embed. Here you enter a label (in column 1!) followed by the word Routine, like this:

```
MyProcess       ROUTINE
```

After this enter, indented at least one character, type the following:

```
  Next(OneFile)
  If ErrorCode()
    Post(Event:CloseWindow)
```

```
    Exit
  End
  Two:Record = One:Record          !Some process
  Add(TwoFile)
  Exit
```

Exit and save. Finally go to Window Events|Timer and insert a Source embed. Here you enter:

```
DO MyProcess
```

Exit and save. Close the embeds and press Ok on the procedure properties screen.

This is the most simplistic translation of the Q&D tight loop above into a timed loop. Yes, although you never typed the word LOOP, it is still there, namely the ACCEPT loop. A nice feature of the ACCEPT loop is that it allows the operating system to do its thing, and it allows windows to refresh. And it is always LOOPing and waiting for some event.

In this case ACCEPT gets a periodic timer event via the TIMER attribute on the window. Every 1/100 of a second the Event:Timer is fired. And when it fires, the MyProcess routine is called. This routine contains the body of the tight loop above, except that the BREAK statement has changed since this is no longer a simple loop. When the code is done processing OneFile it posts an Event:CloseWindow to effectively terminate the ACCEPT loop.

Using a Window procedure this way is as quick and dirty as the original code and is only meant to demonstrate how easy it is to change a tight loop into a timer loop. Let's make it better and add some features to this concept.

Improvements

The example app, delivered with this article, is the Invoice example where I ripped out everything but the All Orders Tree. In this example I'll export this tree into a CSV file.

I added a procedure called ExportAllOrders for which I chose the Window - Generic window handler procedure type. For the Window I picked the ProgressWindow. The required variables for this window are not automatically added this way. Press the [...] ellipsis button for the Data and copy and paste these variables:

```
Progress:Thermometer BYTE
RecordsToProcess     LONG
RecordsProcessed     LONG
PercentProgress      BYTE
LOC:Action           SHORT
```

This procedure processes three files; it uses a derived WindowManager class with a method (local procedure) for each file plus one to update the progress bar. To derive the WindowManager, press the Windows Behaviour button on the procedure properties screen. Go to the Classes tab and tick the Derive checkbox. Now you can add your own methods under the New Class Methods button. You can also create routines to do the same job, but I prefer WindowManager methods.

When I walk down the All Orders Tree I'll first pass a Customer, then its orders and, at the third level, the order's details, so there are three files to be processed, in that order. All these files are going to be processed from the EVENT:Timer embed. Which file is processed is controlled by the LOC:Action variable. Here's the EVENT:Timer embed:

```
1.  IF LOC:Action = 0 THEN CYCLE .
2.  ProgressWindow{PROP:Timer} = 0
3.  CASE LOC:Action
4.  OF 1          !(or eqCustomers)
5.    SELF.ProcesCustomers
6.  OF 2          !(or eqOrders)
7.    SELF.ProcessOrders
8.  OF 3          !(or eqDetail)
9.    SELF.ProcessDetail
10. OF 4          !(or eqFinished)
11.   LOC:Action = 0
12.   POST(EVENT:CloseWindow)
13. END
```

```
14. ProgressWindow{PROP:Timer} = 1
```

In line 1 the code exits when there's no action (not really necessary as nothing will happen).

In line 2 the timer is stopped, to prevent timer events queuing up while processing a file.

Lines 3 through 13 determine what action is to be performed.

In line 14 the timer is switched back on for further processing.

The value of LOC:Action is not important. When an incremental value is used, the above CASE structure could be replaced by an EXECUTE structure. However, using a CASE structure allows for the use of equates which makes the code more readable; you don't have to remember what value of LOC:Action does what. These equates are defined in the Local Data embed.

But what is controlling the LOC:Action variable? Simple: it's the processes themselves.

When this procedure starts up it first checks for the existence of the AllOrders.csv export file. The code is in ThisWindow.Init, before opening the files; if the file exists it is removed..

After the window is opened and initialized, the process variables are primed and the Customer file is setup for processing. From ThisWindow.Init (priority 8500):

```
Progress:Thermometer = 0            !Start progress bar to zero
RecordsToProcess = RECORDS(Customers) !Process all customers
RecordsToProcess += RECORDS(Orders)   !+ Process all orders
RecordsToProcess += RECORDS(Detail)   !+ Process all details
RecordsProcessed = 0                !Will be inc'd for each record
PercentProgress  = 0                !Calc % finished for progressbar
?Progress:UserString{PROP:Text} = 'Exporting All Orders'
!Prepare the Customers file for processing
SET(CUS:KeyCompany)
LOC:Action = 1                      !Timer-Action = 1 (ProcessCustomers)
ProgressWindow{PROP:Timer} = 1      !Start the timer loop
```

If you need to ask for some data from the end-user then set the LOC:Action and the timer both to 0 (zero). Add your entry fields to the window (e.g. to ask for the name of the export-file) and handle them as you normally would. Under the Event:Accepted of the Ok button on your form you embed the last three lines above to start the process.

Because the Windows Timer is activated, the embed in Event:Timer is called 100 times per second. In this Event:Timer embed, the LOC:Action variable determines what happens. Since it is initially set to 1 (or eqCustomers) its CASE structure calls the derived method ProcesCustomers. Here's the code in ThisWindow.ProcessCustomers:

```
1.  IF Access:Customers.Next() <> Level:Benign
2.    LOC:Action = 4  !(or eqFinished)
3.    RETURN
4.  END
5.  SELF.UpdateProgress
6.  !
7.  CSV:Line01 = CLIP(CUS:Company)
8.  CSV:Line02 = CLIP(CUS:FirstName) & ' ' & CLIP(CUS:LastName)
9.  CSV:Line03 = FORMAT(CUS:CustNumber,@P(#######)P)
10. CSV:Line04 = CLIP(CUS:City)
11. IF Access:Export.Insert() <> Level:Benign
12.   RETURN
13. END
14. !
15. !Setup the orders file to process the orders for one customer
16. ORD:CustNumber  = CUS:CustNumber
17. ORD:OrderNumber = 0
18. SET(ORD:KeyCustOrderNumber,ORD:KeyCustOrderNumber)
19. !
```

20. LOC:ACtion = 2 !(or eqOrders)
21. RETURN

At line 1 the next customer is fetched in the order set in ThisWindow.Init. When the last record is read (the .Next() doesn't return Level:Benign) then LOC:Action is set to 4 (or eqFinished) and this method returns to its caller (the timer loop).

At line 5 the ThisWindow.UpdateProgress method is called to update the progressbar.

At line 7-10 the fields of the export file are primed in the same way as in the All Orders Tree code.

At line 11 a record is added to the export file. When this fails, the code skips processing this customer and returns to the caller

At line 16-18 the Order file's key fields are primed and the process order is set.

At line 20 the LOC:Action is set to 2 (or eqOrders) to tell the timer loop to process the Orders file.

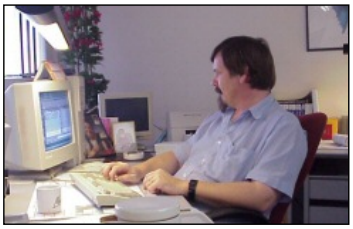
The derived methods: ProcessOrders and ProcessDetail are written in the same way.

Summary

You can process files in tight loops but you won't be able to update the progress bar and you run the risk of making the computer unresponsive for a noticeable period of time. A much better way is to process your file(s) within a timer loop that does not have these disadvantages and works (almost) as fast.

In this article I've shown that it is really simple to build a timer loop. In fact, I prefer this approach over the process template for all my process-oriented coding such as importing/exporting/converting files, generating a Word document and executing queries from a SQL backend.

Download the source



Maarten Veenstra began his computing career as a field service engineer on the DEC PDP11 family and a microprocessor-controlled punchcard machine. His first personal computer job was fixing CP/M computers and the early IBM clones, during which time he bought an MSX "computer" and taught himself BASIC and assembler. Maarten began using Clarion Professional Developer in 1988, wrote his first Windows program with CW 2.0, and his first ABC program with C4. He is now the co-owner of Nepucon, which provides service to public utilities. Maarten married in 1991, and he and his wife have adopted two beautiful Chinese girls.

Reader Comments

Posted on Thursday, January 31, 2008 by John Cortenbach

Interesting solution. Assuming I use the LOOP contruct as you show in your article, YIELD give my system great reponsive. (allowing for progress bar updates, switching between windows and coming back to an active screen that is still updating, etc., etc.)

What do you see as the pros and cons of that approach?

Posted on Friday, February 01, 2008 by M Veenstra

John,
I use YIELD in one-time conversion programs which are mostly handcoded and optimised for speed. The YIELD keeps the "busy" window visable. SLEEP is also a nice way to give timer-slices back to the OS.
Timer based loops are OS friendly. It leaves the OS in total control over the devison of the process-time-slots.
In Clarion it has the advantage that you can use the available templates (and therefor also any 3rdparties, if needed). This means that most of the coded needed is generated. Deriving the windowmanager allows you to write well structured and easy maintainable code. And with only a few lines of code I have 'emulated' the process template but with much greater flexebility.
Regards, Maarten.

Posted on Thursday, February 07, 2008 by Rudolf Kreutzer

Maarten,
thank you for this nice example, it makes it very easy to understand how the timer-event works.

One important thing that you forget to tell is, that the maximum speed of your example is exactly 100 records (loops) per second. This is very less and slow. If you have to run thru 100.000 records the programm needs a minimum of 16 minutes!

So it is a good idea to surround the timer-code with an extra "loop xxx times" to run xxx records at once.

Posted on Friday, February 08, 2008 by M Veenstra

Rudolf,

Yes, you are right. I should have mentioned that. Depending on the file being processed I do a 'loop 25 tiems' or use a variable for that. In another situation, where I know there are a limited # of records (< 100) for a child file, I process these at once.

Thanks for adding this important info to this article.

Regards, Maarten

Add a comment

Clarion Magazine

Create a Report using MS SQL Views

by Robert Johnson

Published 2008-01-22

I had need of a report that required data from several MS SQL tables that were not related (nor could be for technical reasons); that made creation of this report an interesting challenge in Clarion. The report just wouldn't output the data correctly no matter what I did. I tried everything, filters, keys, the whole ball of wax, but no joy.

Then it hit me. Views! SQL views are virtual tables made up of selected data from one or more actual tables. Why not create a view on the server that would make all the necessary joins for me? The view could give me the output I needed all without my having to write a single line of code. This is SQL at its best and what it's made for, I thought, so firing up SQL Server Management Studio I got to work to create my view.

The report

The report I needed had to retrieve data from five tables. Three were related to each other, and the other two were related to each other, but no tables in the group of three were related to the group of two (see Figure 2). I knew that I could define a view by simply by dragging and dropping between tables on the graphic diagram, and SQL Server Management Studio would write the SQL code for me. Here's how it's done.

Step One: Create the view

First, I right-click on the View folder under my database and select New View, which pops up the Add Table dialog box (Figure 1). I select the tables I want for my view.

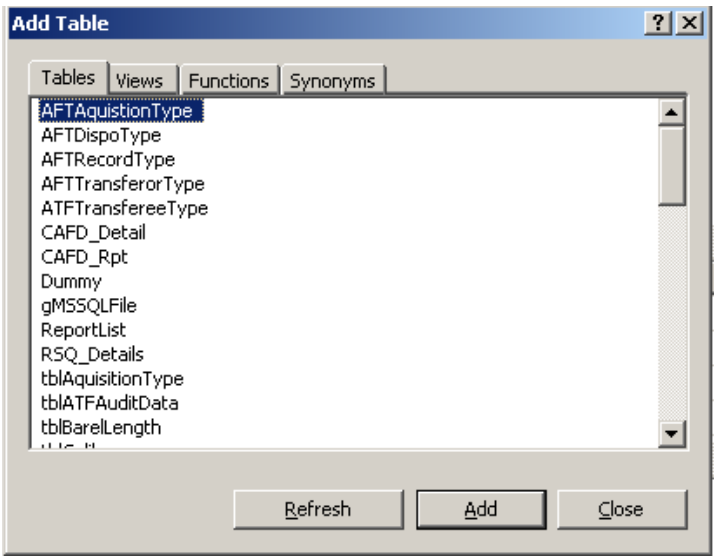


Figure 1. Adding a table in SQL Server Management Studio

After I select my tables the screen looks like Figure 2.

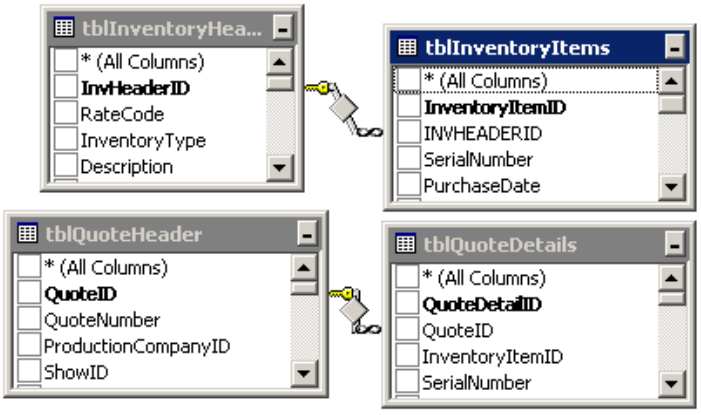


Figure 2. Main design view

The top portion of Figure 2 shows the graphic representation of the tables and the relationship between the tables. Next down is an area where I can select columns I want in my view, and below that is the SQL code generated based on my actions. Notice that there is no relationship between tblQuoteDetails and tblInventoryItems. In a normalized database I realize that these tables should be related, however because of restrictions placed on me I was not able to relate these two tables. However in the view I can relate to my heart's content. All I have to do is drag from tblQuoteDetails InventoryItemID to tblInventoyItems InventoryItemID and "Voila, she's a related" (Figure 3).

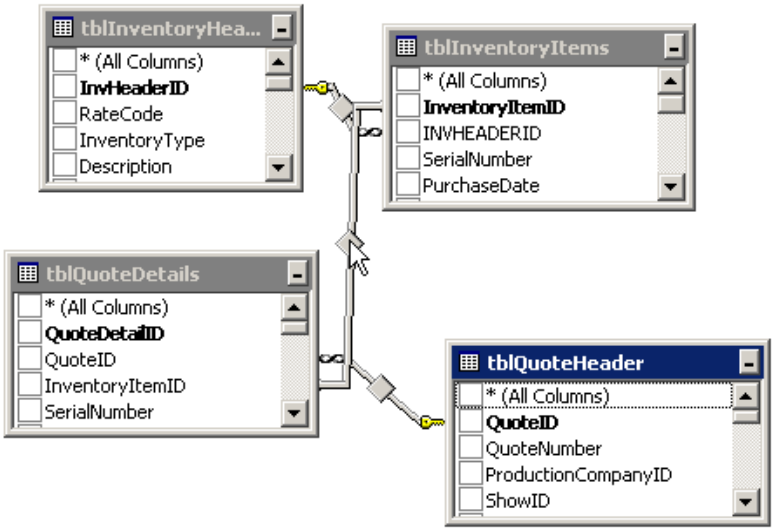


Figure 3. After new join

You can now see that I have a relationship between tblQuoteDetails and tbInventoryItems where none existed before. Now all I have to do is to select the columns I want in my report (Figure 4).

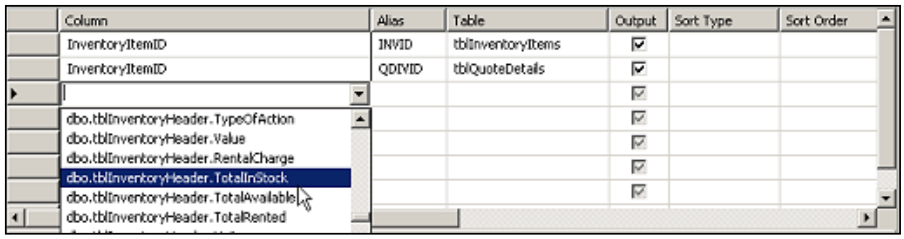


Figure 4. Adding columns to view (view full size image**)**

The pull down combo box in the first column lets me choose from a list of all available columns from the selected tables. I can enter an Alias for each column here, something I find it helpful when I work in Clarion. One note: a column's alias cannot be the same as the column name; it has to be something different. You can enter Sort Type and Sort Order criteria here also. Notice that as you enter columns, the code below updates to reflect your changes.

After I finish adding all my required columns I save the view. Testing the output with a query is easy. After saving the view, I right click on the view and select Script View as | SELECT TO | New Query Editor Window (see Figure 5). This creates a partial query listing the elements of the view. All I needed to do to complete the query was add a Where clause to match my report's requirements (see Figure 6).

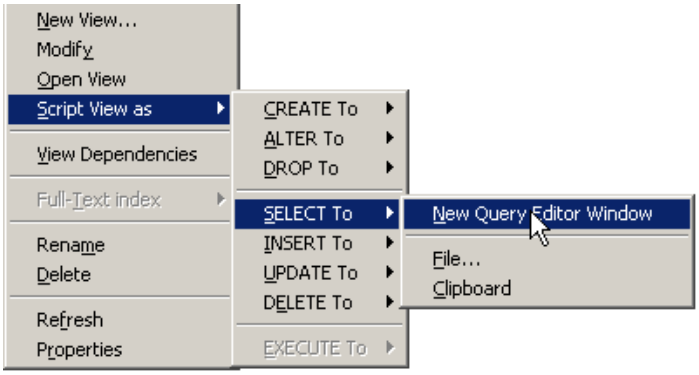


Figure 5. Menu for test query

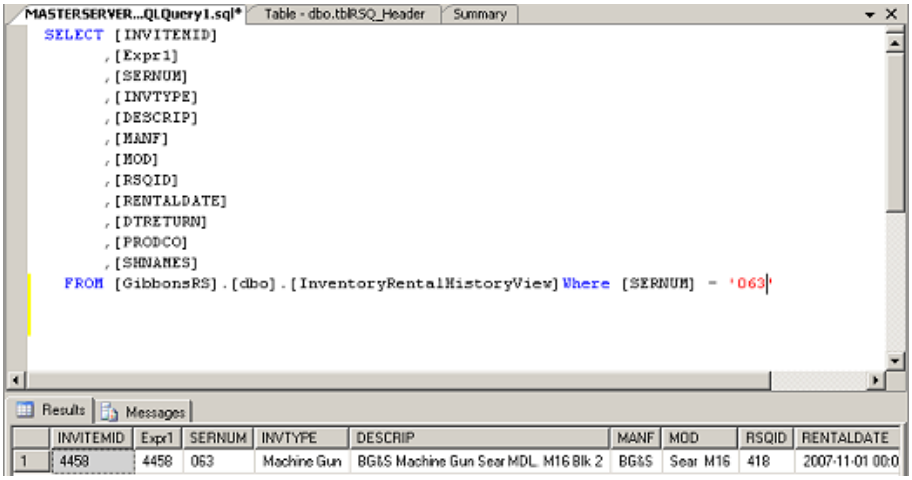


Figure 6. Test query and output (view full size image**)**

As you the bottom of Figure 6 shows, my test output was successful and was just as I expected.

Step Two: Define the view in the dictionary

The next step is to define the view in the Clarion dictionary. With the dictionary open I choose File | Import Table (Figure 7). From the resulting dialog box I select the MSSQL driver (Figure 7).

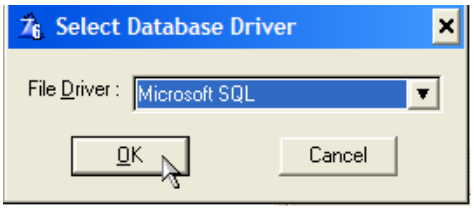


Figure 7. Select driver dialog box

The next dialog is the server login screen (Figure 8); I complete the required fields and login to my MSSQL server. Figure 9 shows a list of tables; I select my newly created view and press the Finish button to complete the Import.

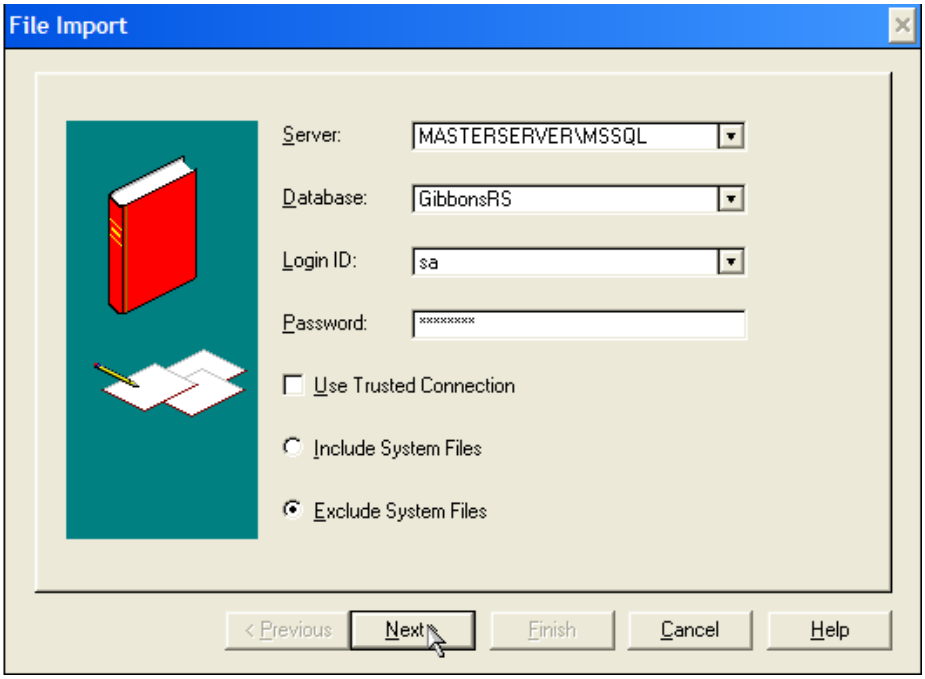


Figure 8. Server login

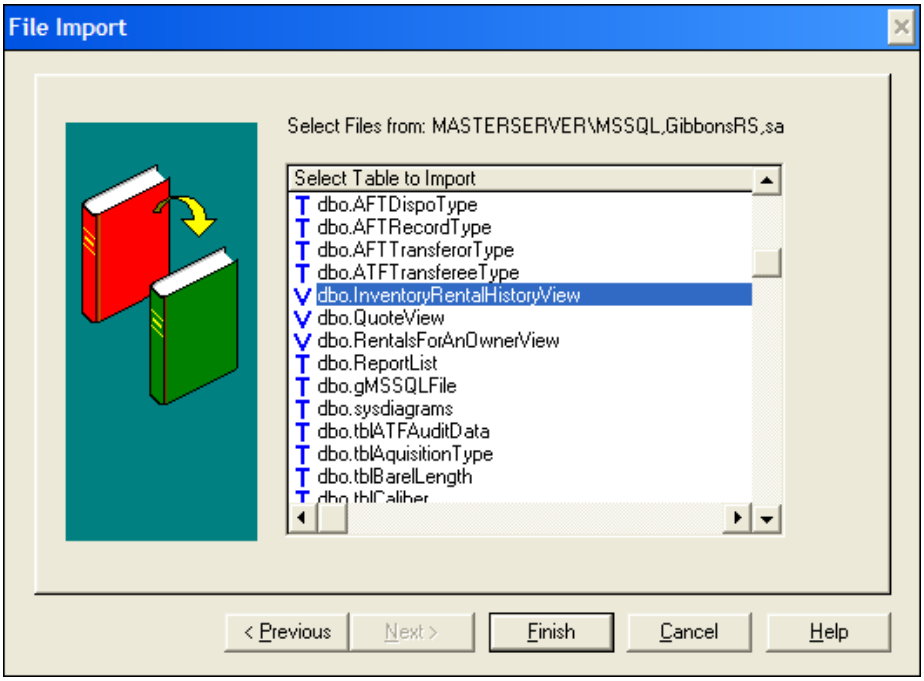


Figure 9. Select view to import

All I have to do now is add a key to my view. Figures 10-12 show the definiton of a simple key on the serial number with no Auto Number or Unique attributes required.

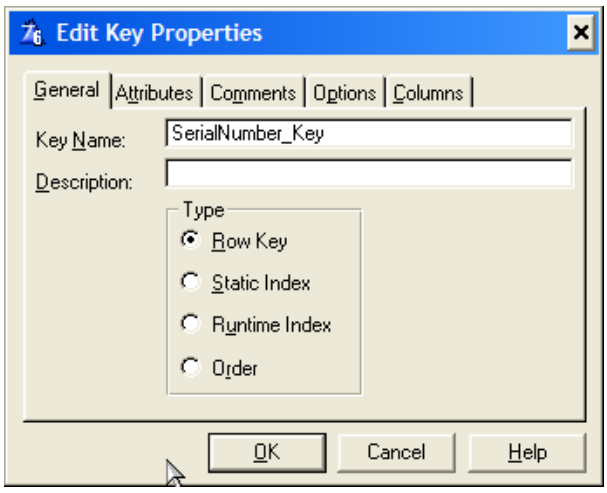


Figure 10. Key attributes

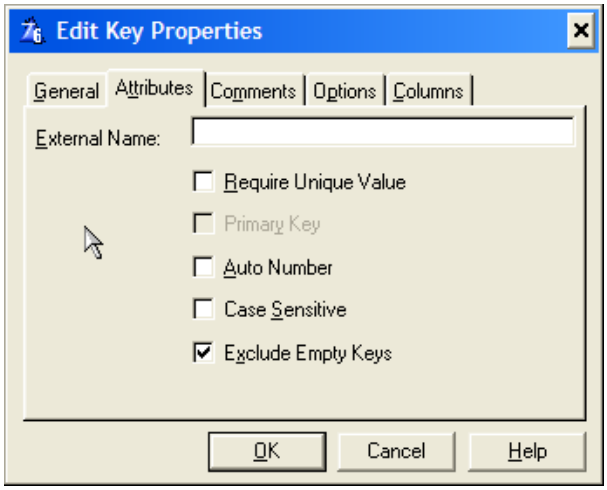


Figure 11. Select key column

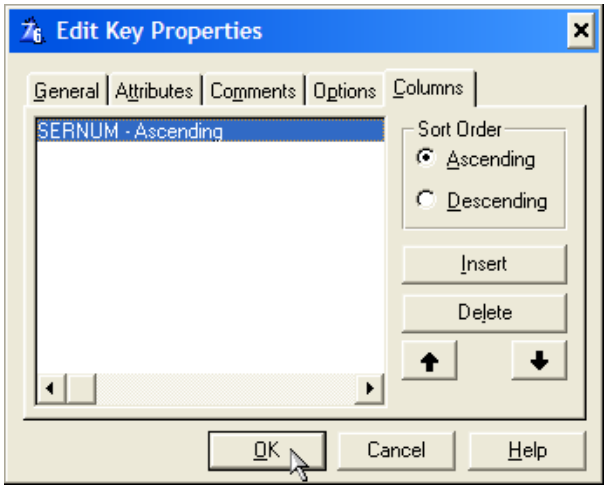


Figure 12. Set key column

Step Three: Create the report

Now I can close and save the dictionary. I open my application in Clarion and create the report using the standard Clarion report generator or any third party product. I specify my view as the table and select the key I've defined, and I am in business. From this point on, its business as usual.

Summary

Sometimes managing table relationships within a Clarion report is too difficult and doesn't yield the desired results. If you're using MS SQL (or any other SQL database that supports views) you can always define your view on the server, import the view into your dictionary as a table, and use the view to drive the report.

Reader Comments

Posted on Monday, January 28, 2008 by Terry Hill

Thanks for the good article. One question.

When I create keys for MS-SQL tables in the dictionary, I always select the Case-Sensitive attribute. Is this recommeded when creating a key for a view as well?

.....

Posted on Monday, January 28, 2008 by Robert Johnson

Terry, you are correct regarding Case Sensitivity in normal usage however in a View situation I honestly don't know the answer to that one. In my case performance was satisfactory in all cases so I didn't set the flag. I need to go back and run some tests to see if setting the flag improves performance. Thanks for noticing and bringing it to my attention.

Robert

Add a comment

Clarion Magazine

Capturing Standard Output From A Console Program

by Rick Martin

Published 2008-01-17

In a previous article I described a string class that handles strings of arbitrary length. I originally developed the string class to solve a specific problem: I was working on the tools for a build system that integrated Clarion and CVS version control and I needed to capture the output from CVS commands for logging purposes. That led me to an API-based solution that neatly captures console output using pipes. In this article I'll show you how you can do the same.

The requirement

At the time I wrote the CVS integration code, I was working for a publicly held company. In the United States public companies have to comply with the Sarbanes-Oxley Act of 2002 (SOX). SOX has a big impact on the process and procedures for IT departments. One requirement was that we only build modules that actually changed for each release of our internal software; I had to be able to document which modules had changed for the SOX auditors.

The rdiff command in CVS returns all of the modules that have changes between different tags, dates or branches.

The command below returns a list of every file in the MyProgram project that has changed between the tags Version_2_01_01 and Version_2_01_02.

```
cvs -q rdiff -s -rVersion_2_01_01 -rVersion_2_01_02 MyProgram
```

The output from rdiff looks like this:

```
File ChangeLog,v changed from revision 1.52.2.5 to 1.52.2.6
File foo.c,v changed from revision 1.52.2.3 to 1.52.2.4
File bar.h,v changed from revision 1.29.2.1 to 1.2
```

For the project I was working on there were about 4000 modules, and I had no way to know in advance how long the output would be from the CVS command.

I originally thought I would use the RUN command to redirect the output from CVS to a text file and then read the file to get the contents. This worked but looked unprofessional because the command window appeared while the CVS command executed.

I had a class that used the CreateProcess API call to start an external program, and it included options to hide the console window. I believe the class was originally based on the code from Power Run. This class made it easy to hide the command window, but I still thought I could do better than using file redirection to capture the results. Poking around on the newsgroups I found a post from John Christ that demonstrated using CreateProcess to redirect output to a file rather than using > MyOutput.Txt in the command line.

One of the structures passed to CreateProcess is the StartupInfo group:

```
api_StartupInfo    GROUP,TYPE
cb                 api_DWORD
lpReserved         ULONG
lpDesktop          ULONG
```

```
lpTitle             ULONG
dwX                 api_DWORD
dwY                 api_DWORD
dwXSize             api_DWORD
dwYSize             api_DWORD
dwXCountChars       api_DWORD
dwYCountChars       api_DWORD
dwFillAttribute     api_DWORD
dwFlags             api_DWORD
wShowWindow         USHORT
cbReserved2         USHORT
lpReserved2         ULONG
hStdInput           api_HANDLE
hStdOutput          api_HANDLE
hStdError           api_HANDLE
                END
```

According to MSDN, if the constant STARTF_USESTDHANDLES is BANDed (Binary AND) into dwFlags, CreateProcess will use the api_Handle values specified in the StartupInfo group. If you create a handle to a file and pass it in the hStdOutput and hStdError fields, CreateProcess automatically redirects any output from the process to that file.

This approach worked; I launched the cvs command via CreateProcess, which created a text file with the CVS output. I could then read the file using the ASCII driver after the call finished. However, I don't like to leave things alone if I think there is a better solution. I knew that other tools, like WinCVS, were communicating directly with CVS without using temporary files. So I dug around on MSDN in the pages related to CreateProcess until I found an example using CreatePipe to directly capture standard output and error inside of code. Ah hah! I had found what I wanted.

A pipe in Windows is a lot like a real pipe. Information is poured into one end of the pipe by a process and the information is received at the other end by another process. The key was to create a pipe for the child process, and then specify that pipe for standard output instead of a file handle. That way the child process "writes" the information into the pipe which is redirected back to the calling process.

Unfortunately, as with many Windows API tasks, it was not as simple as it first sounded. Here are a couple of gotchas:

- A pipe has two ends. One is written to and the other is read from. As it turns out, if you want a child process to write to a pipe created by the parent you have to close the write end of the pipe in the parent process before calling the child. It's a rule I didn't try to argue with. Basically, your program hangs if you don't do it.
- Another thing I wanted was to capture both standard output and error into the same text stream, so I needed a way to tell the child process to send both output streams to the same pipe. Fortunately, this wasn't too difficult, and I'll show how it's done.
- Finally, when you create a pipe you have to tell the OS that the pipe is inheritable by child processes.

The code to create a pipe looks like this (where saAttr is an instance of the Windows API SECURITY_ATTRIBUTES structure):

```
CLEAR(saAttr)
saAttr.bInheritHandle = TRUE
saAttr.nLength = SIZE(saAttr)
IF NOT api_CreatePipe(ADDRESS(hChildStdoutRd), |
            ADDRESS(hChildStdoutWr), |
            ADDRESS(saAttr), |
            0)
```

```
    Self.ErrorHandler()
  END
```

This creates the pipe and specifies that child processes can inherit the pipe. hChildStdoutRd and hChildstdoutWr are handles to the read and write ends of the pipe.

Now I want to create a duplicate handle for the pipe to use as standard error. To do this I obtain the handle to the current process, and then call DuplicateHandle to create the standard error pipe handle.

```
  hProc = api_GetCurrentProcess()
  ! create an inheritable duplicate handle to the
  ! pipe to use for the stdErr value
  IF NOT api_DuplicateHandle(|
              hProc, |
              hChildStdoutWr, |
              hProc, |
              ADDRESS(hChildStdErrWr), |
              0, |
              TRUE, |
              api_DUPLICATE_SAME_ACCESS)
    Self.ErrorHandler()
  END
```

Note that I am passing the handle (hChildStdoutWr) returned from the CreatePipe call into DuplicateHandle.

Now I have a pipe that can be used both for standard output and error (hChildStdoutWr and hChildStdErrWr) for my child process.

Next I need to create a non-inheritable handle to the pipe for the parent class to use when reading the output from the child process.

```
  ! create a NON inheritable duplicate handle to the
  ! pipe to use in this process for reading the pipe
  IF NOT api_DuplicateHandle(hProc, |
              hChildStdoutRd, |
              hProc,|
              ADDRESS(duphChildStdoutRd), |! Address of new handle.
              0, |
              FALSE,          |! Make it uninheritable.
              api_DUPLICATE_SAME_ACCESS)
    Self.ErrorHandler()
  END
  ! close the Child read end of the pipe for stdout and
  ! stderr. The child process will only write to the pipe.
  IF NOT api_CloseHandle(hChildStdoutRd)
    Self.ErrorHandler()
  END
```

Note that I am closing the read end of the inheritable handle of the pipe I am passing to the child. The child will only write to the pipe, and leaving the read end of pipe open causes the process to lock up.

Now that I have my pipe properly created I set the standard output and error variables in the StartupInfo structure.

```
!Set the process startup info handles to the new write handle for our pipe
ExecCmdStartupInfo.hStdOutput = hChildStdoutWr
ExecCmdStartupInfo.hStdError  = hChildStdErrWr
```

Everything I need is now in place I call CreateProcess to kick off the child process. Right after calling CreateProcess I need to close the write ends of the pipe in the parent so the child process can write to it. The child process has created its own handles to the pipe at this point so there is no harm in closing them in the parent. Again, if I don't do this the program will hang.

```
IF NOT api_CloseHandle(hChildStdoutWr)
   Self.ErrorHandler()
END
IF NOT api_CloseHandle(hChildStdErrWr)
   Self.ErrorHandler()
END
```

Now I open a Clarion window (defaulted to hidden) with a timer and use the ReadFile API call on the pipe to receive the data from the child process. The data is not returned in one big block. Instead it is returned in smaller chucks as the child process writes it; so I have to handle receiving the output in slices and appending it together to capture the entire thing. And that, if you'll recall, is why I wrote StringClass.

Once the child process terminates I break out of the window and clean up all of the handles.

All of the above is a bunch of semi-complicated stuff that you don't have to worry about at all, as I've encapsulated the code in an easy-to-use class. All you have to do to capture the output of program is type and run the following code:

```
RunCommand         CaptStdOutCL
ResultsString      StringClass


 CODE
 RunCommand.CreateProcessCaptureOutput(|
  'cvs -H checkout', api_SW_HIDE, false,,,,,ResultsString)
```

After this, ResultsString holds the output from the command.

The full parameter list for CreateProcessCaptureOutput is as follows:

```
CreateProcessCaptureOutput  PROCEDURE (|
        STRING pExecCmd, |
        USHORT pExecMode, |
        BYTE pUseCMD=1, |
        <STRING pPath>, |
        <STRING pMessage>, |
        <STRING pTitle>, |
        <*LONG pExitCommand>, |
        <*StringClass pResults>),LONG,PROC ,VIRTUAL
```

- pExecCmd - The command to execute including any parameters to the program to call.
- pExecMode - The execute mode. This lets you control a number of options for the child process including whether or not the command window displays (see MSDN for a full list).
- pUseCMD - If true then COMSPEC is read to get the name of the command program and the child process is invoked

using this value.

- pPath - If passed the child process is started in the specified directory.
- pMessage - If passed the window inside the class is displayed and pMessage is assigned to the string control on the window.
- pTitle - If passed the window inside the class is displayed and pTitle is assigned to the window title.
- pExitCommand - Returns the ErrorLevel or exit code from the child process.
- pResults - Returns standard output and error from the child process.
- Return value: If the function succeeds in calling the child process then true is returned.

Note on API usage:

You may have noticed that the API calls, constants and structures in the example code all have an api_ prefix. I included a file with the example code that prototypes a lot of the Windows API. There isn't a single source in Clarion that prototypes everything in the Windows API and I was constantly running to duplicate symbol warnings or missing prototypes. I created WinAPI_EQU.INC to hold all of the structure definitions, constants and prototypes I use. I prepended api_ to avoid conflicts with svapi.inc and third party utilities that also create Windows API definitions. You are welcome to use WinAPI_EQU.INC in your programs.

The example code includes the CaptStdOut class, the StringClass source, a very simple project to demonstrate the class in both Clarion 6 and 7, and a template that includes the class in either an ABC or Clarion template chain APP. If you are going to use the class and template in an ABC APP project then you need to put the class INC and CLW files in the LibSrc folder and add the template to any APP that will use CaptStdOut, as well as to the data DLL if you are working with an ABC-based multi-DLL project.

Download the source

Rick Martin has been programming systems and applications for over 20 years. Outside of work Rick avoids high-tech as much as possible and enjoys hobbies in golf, woodworking, and the outdoors. He and his wife Cathi are enjoying their new grandson who lives with his mom just minutes away. Rick makes his home in beautiful Chico, California.

Reader Comments

Add a comment

Clarion Magazine

Querying ActiveDirectory In Clarion

by Marty Honea

Published 2008-01-17

Software development is continually evolving. In the short time I've been in the business, we've come all the way from Windows 3.1 to Windows Vista, and one of the biggest changes has been the growth of Windows networking. As with other areas of the Windows OS, as networking has progressed, the management tools have become more powerful and, hopefully, easier to use. Active Directory is one of those tools.

Until recently, I knew how to work with Active Directory, but avoided it whenever possible. All of that changed a few days ago when a client called and asked "How much do you know about Active Directory?" They needed the ability to tie the login for a particular program to Active Directory, and prevent access if the Active Directory login credentials had expired. As I would find out later, this is one of the simplest things to do in Active Directory, and only the tip of the iceberg when looking at the information available. Using Active Directory, my client could have just as easily limited the login for a particular person to a single PC and given them a window of time during which they were allowed to login.

Google is my friend

As with most things that I'm unsure of where to start, I turn to Google for a starting point. Google, my trusted friend, once again gave me all that I needed to know about Active Directory, LDAP, Scripting, and dealing with these things in every language in the world except for Clarion.

After reading way too much on what was possible with Active Directory, I realized that if I wanted to programmatically get anything from Active Directory I would have to learn a new way of communicating. Here is where LDAP entered the picture.

LDAP is defined in Wikipedia as:

> The Lightweight Directory Access Protocol, or LDAP (IPA: [ˈɛl dæp]), is an application protocol for querying and modifying directory services running over TCP/IP.

Directory Services are defined as:

> "A directory service (DS) is a software application — or a set of applications — that stores and organizes information about a computer network's users and network resources, and that allows network administrators to manage users' access to the resources. Additionally, directory services act as an abstraction layer between users and shared resources."

Directory service sounds exactly like Active Directory (as it should, because Active Directory is an example of a directory service), and that means that LDAP must be the way to go to communicate with Active Directory.

If in doubt, ask in the newsgroups

Clarion has a bunch of things going for it. If you ask ten Clarion developers what is the best thing about Clarion, you'll probably get ten different answers. And if those ten developers are anything like me, you're probably not going to get the same answer two days in a row. On the day I decided to learn how to do LDAP in Clarion, if you had asked me that question, I would have told you Clarion's best feature is the newsgroups on the discuss.softvelocity.com news server.

And on this day in particular, Bjarne Havnen was pretty high up on my list of helpful developers. Two hours and fifteen minutes after I asked in the Comp.Lang.Clarion newsgroup if anyone had worked with Active Directory, Bjarne had not only answered but he had some source code for me to start with. So if you get any benefit out of this article, be sure and tell Bjarne thanks too!

ADO in Clarion

It turns out that you can execute LDAP queries in Clarion with the little-known Clarion ADO class. Bjarne's help and a Clarion Magazine article by Tom Ruby on SoftVelocity's little-known Clarion ADO wrapper class got me most of the way there.

To add the ADO class to an application, all I had to do was include the cwado.inc file in my application and add the project defines listed below.

```
_COMLinkMode_
_COMDLLMode_=>off
_ADOLinkMode_
_ADODLLMode_=>off
_svLinkMode_
_svDLLMode_=>off
_ADOMPRLinkMode_
_ADOMPRDLLMode_=>off
_SVDLLMode_=>0
_SVLinkMode_=>1
_ABCDllMode_=>0
_ABCLinkMode_=>1
```

After these were added, I was ready to start coding my procedure.

In my data section I added:

```
COMIniter    CCOMIniter
dbConn       &cConnection
szConnectStr Cstring(255)
HR           HResult
```

In my data section I added:

```
resultq                QUEUE,PRE()
! Fields in the queue
                       END
```

This gave me everything I needed to use ADO in my procedure, and a place to put the data I got back from my query. Now I had to write the code to actually use it. And with Clarion 6.x it only took a few lines of code to initialize everything:.

```
szConnectStr ='Provider=ADsDSOObject'
If COMIniter.IsInitialised()
  dbConn &= new(cConnection)
  If ~(dbConn &= null)
    hr = dbConn.Init()
```

```
    If hr = S_OK
      hr = dbConn.Connect(szConnectStr)
```

At this point, I've told the ADO class what type of connection I'm making; I've initialized the class, created and initialized a new connection, and I've made the connection to the Active Directory Service. That's a lot of stuff to do in seven lines of code, and two of those lines are error checking!

Now I'm ready to actually use the connection; for this I've made a routine named readperson. I'll also wrap up the error checking lines of code and when I'm done, I'll close the connection and dispose of the connection.

```
      If hr = S_OK
        do readperson
      else
        !Something went wrong opening the connection
      end
    end
    hr = dbConn.Close()
    If hr = S_OK
      !Connection closed ok
    else
      !Connection didn't close
    end
    dispose(dbConn)
  else
    !Instance creation went wrong
  End
END
```

The error handling in this code is nothing more than commented lines to let you know what to check for at that point. What I do for an error and what you do for an error will be totally different. You'll have to salt to taste with your error handling.

Next comes the readperson routine that I'm calling after opening the connection.

For this example, I'll retrieve a list of users with expired accounts.

```
ReadPerson routine
 Data
Rows Long !
options long
_Hr Hresult
sqlStr Cstring(255)
Rs &CRecordSEt
bEof short
Mapper &TableMapper
 Code
   SqlStr = '<<LDAP://DC=YourDomain,DC=com>;(&' |
   &'(objectCategory=person)(objectClass=user)' |
   &'(userAccountControl:1.2.840.113556.1.4.803:=2));uid,' |
   &'sAMAccountName,cn,sn,name,name,givenname,title,description,' |
   &'department,legacyExchangeDN,telephoneNumber,mail,lastLogon,' |
```

```
&'lastlogoff,logonhours,pwdlastset,accountexpires;subtree'
```

This string looks funny, but that's because it's in the LDAP query language structure. I'm not going to go into a lot of detail on the language structure. There are a lot of resources out there that can do a better job explaining it than I can, and it's outside the scope of this article. But two good resources I would recommend are both on the Microsoft website.

- Reading User Account Password Attributes
- Active Directory Script Repository

I will talk about two things in the string. The first is the section that starts with <<LDAP://DC. This string lets the system know which Active Directory Domain you want to query. So you'll need to change the "YourDomain" to the name of your domain. The second is the strange section (userAccountControl:1.2.840.113556.1.4.803:=2). This section tells the scripts to search for objects (in this case, users) where bit 2 in the userAccountControl attribute has been enabled. I won't go into any great detail on the attributes, but for this example just know that if bit 2 is enabled the account is disabled. So, that brings up the question of why didn't they just use (userAccountControl=2)? Well, 1.2.840.1113556.1.4.803 in LDAP is a bit matching rule, and is equivilant to the Boolean AND operator. In Clarion the code would be:

```
BAND(userAccountControl,0010b)
```

If you're familiar with bitmasks this might make some sense to you. If not, well, don't worry too much about it, it doesn't make a lot of sense to me either.

On to the rest of the code; I'm passing in the query and checking to see if there was an error here.

```
rs&=dbConn._Execute(SqlStr,Rows,options,_Hr)
if _Hr=S_OK                    !alt ok
```

Next, I'll loop through the records returned and put them in my resultq. If I get an end of file (EOF) error, I'll break out of the loop. TableMapper is an ADO helper class (included with the SV ADO code) that translates ADO result sets to Clarion GROUPs and QUEUE records.

```
  Mapper &=new(TableMapper)
  loop
    hr = rs.GetEof(bEof)
    if hr = S_OK
      if bEof = -1     ! with ADO, true is -1 and false is 0
        break          ! break the loop
      else
        Mapper.MapRsToGroup(Rs,resultq)
        add(resultq)
      end
    end
    hr = rs.MoveNext()
    if hr = S_OK
    else
      break
    end
  end
  dispose(mapper)
Else
```

```
    !LDAP Failed
  End
  Exit
```

After loading the queue, I dispose of the Mapper instance, and exit the routine.

At this point, I've got the data I need in my queue, and I can handle them however I want in my program. In the example program I've just displayed them in a list box, but in real life I'd be more likely to be looking for a specific account and validating the login in my program against Active Directory.

Using this same LDAP method, and just modifying the script a little, I can not only query the Active Directory Service, but I can ask it to modify accounts. See Microsoft's script repository page for a wide variety of scripts for just about any kind of interaction you may find yourself needing to do.

Summary

Active Directory is a powerful tool. It gives you the ability to find out about almost anything on your network. I've just scratched the surface of what can be done in this article, but it should be enough to get you thinking and to point you in the right direction when accessing Active Directory.

Download the source

Marty Honea wrote his first program at a summer camp when he was nine, in BASIC on an Apple IIe. Among other jobs, he has worked as a draftsman, as an Emergency Medical Technician, and as an IT manager for three prisons. He picked up Clarion for Windows in 1998 and, despite having no formal training in computer programming, has managed to make a living with Clarion ever since.

Reader Comments

Add a comment

Clarion Magazine

Lists, CHOICE, and Hidden Tabs

Published 2008-01-11

The main screen of my batch compiler, GTL, has always bothered me.

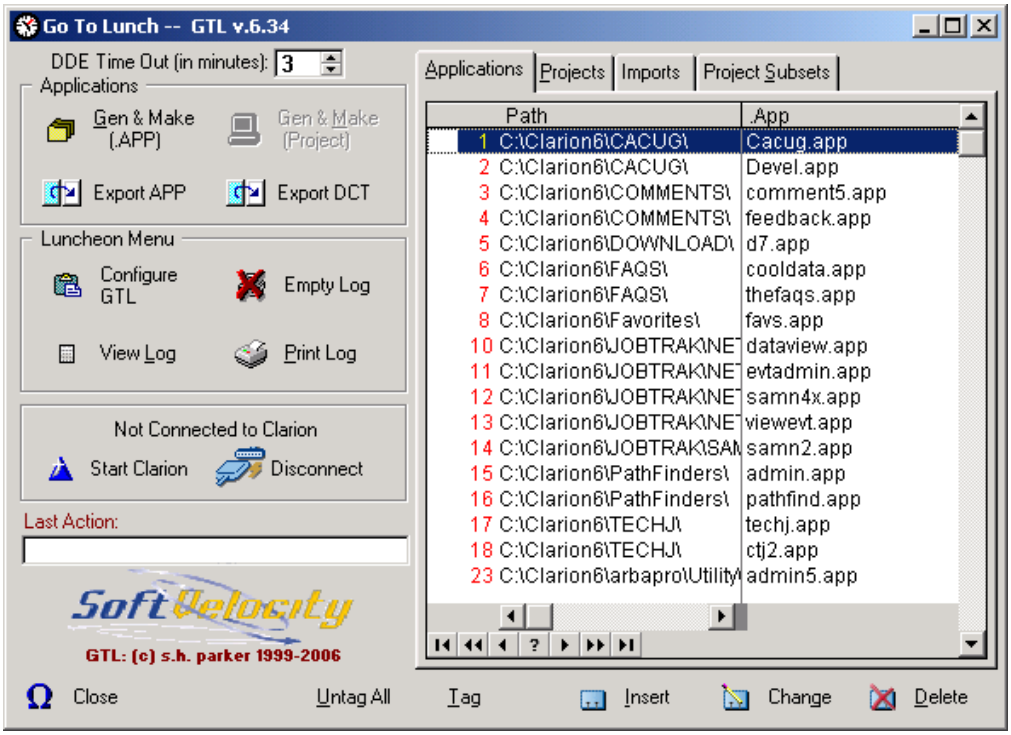


Figure 1. GTL's main screen

So many buttons. So little screen-estate.

GTL didn't start out this way. At first, there were only "Gen & Make" and the two "Export" buttons. There were no tabs. There was lots of screen-estate. I filled some of the empty space with graphics on the thesis that I'd paid for that screen-estate and I was going to use it.

Then came the requests - never let it be said that Clarion developers do not try to get maximum value for their money (Editor's note: GTL is free). Tabs and buttons multiplied as I wedged various new pieces into the available space on the existing window. "Feature creep," a phenomenon with which I am sure Clarion Magazine readers are familiar, eventually led to a screen with almost no space for new buttons (read "features").

Then Microsoft abandoned the tab interface. In fact, Microsoft abandoned the entire MDI interface - it is not thread safe. Yes, Microsoft did recently re-adopt tabs, but as a replacement for toolbars. Indeed, tabbed browses ended up being cited on some "interface hall of shame" website (much to the amusement of most of us on the Softvelocity news groups). So the GTL user interface, in addition to being a bit busy, is also "officially" passé. But how to bring it up to date?

Choice shows the way

Recently I was having problems with the Choose statement (see Using CHOOSE With PROP:SQL). I often confuse Choose and Choice. Choose got me thinking about Choice. It struck me that Choice and sheet controls have a rather important feature in common.

Choice "Returns a user selection number." Most often, I use Choice in a list control to load the highlighted queue record. For example:

```
Get(myQueue, Choice(?List47))
```

In this case, Choice(?List47) returns a pointer to the list box. The Get then retrieves the matching queue entry.

Tabs

Similarly, I determine which tab on a sheet is active, not by the tab's field equate value, not by its label but by its ordinal number (though the field equate should, in fact, resolve to this number also). For example:

```
Case Choice(?CurrentTab)
Of 1
  ?Browse:1{PROP:Format} = |
      '50L(2)|FM~Number~@s16@#1#'        &|
      '62L(2)|FM~Disc. Group~@s20@#2#'   &|
  ! etc.
```

(In this case, I am reformatting a browse to show the sort field in the first column as described in the on-line help.)

And, there it is:

A. Choice returns a number
B. Tabs are identified by their number

Therefore:

C. I can use Choice to get a number and use that number to select a tab!

But didn't I just say that Microsoft had abandoned tabs? What if it were possible to use tabs without *seeing* tabs?

Figure 1, above, has four tabs. Suppose I had a list box formatted to show similar choices (see Figure 2 - in this case, the list is populated without the control template and I can either name my own queue or supply the queue elements, as shown).

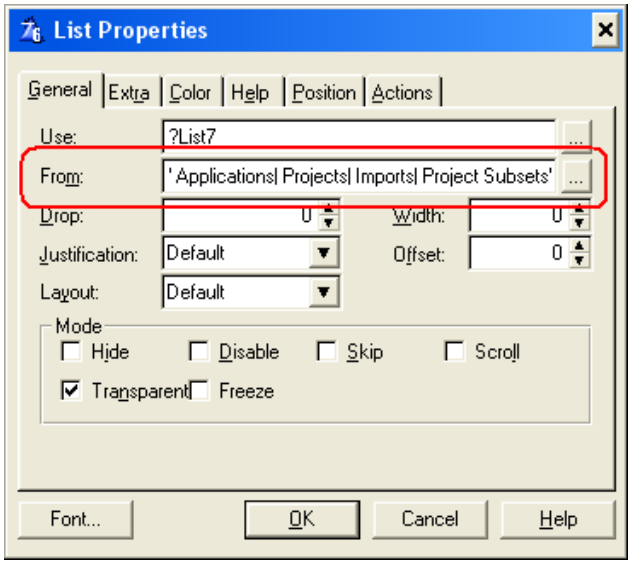


Figure 2. Formatting a list like my tabs

Then, in the list box's Accepted embed:

```
?Sheet1{Prop:Selected} = Choice(?List7)
Post(Event:NewSelection,?Sheet1)
```

Because this small list would allow the end user to "switch tabs" without having to click on an actual tab, I can hide the tabs. I can also hide the sheet. In this way the user interface can be much cleaner. I can also use the screen-estate previously occupied by the tabs to make the list boxes taller, showing more functional information.

In the sample app, downloadable at the end of this article, select Changing Tabs from the main menu. Note: this app was created with 9056. The EXE is compiled in local mode so that users of all Clarion versions can run it.

Menus

The Luncheon Menu group (center left in Figure 1) contains four buttons. Each button calls a procedure.

I can use exactly the same technique as I used to change tabs to call procedures. This allows me to replace the buttons in the Luncheon Menu group with a small list.

I could create a local queue and use it to populate the From field in the List Properties worksheet. Or, I can list the choices, as in Figure 2 above, letting the window formatter set up the From data. Because in this case, as in the previous case, there are few options and they rarely change, hard coding them has no real drawbacks. (The important thing is to populate the list control *without* the template.)

So, suppose I have a list with the same options as the buttons currently in the group. In this list's accepted embed, I can call any procedure I like in exactly the same way I switched tabs, simply by knowing the Choice(?List):

```
Case Choice(?List2)
Of 1    ! Configure GTL
  ConfigureGTL
Of 2    ! Empty Log
   EmptyLog
Of 3    ! View Log
```

```
    ViewLog
  Of 4    ! Print log
    PrintLog
  Of 5
    Run(ClarionStartString)
  End
```

In the demo app, select Call A Procedure on the main menu.

Aesthetics

These local lists aren't very attractive. Make the list control flat and transparent (choose Prettier on the demo app's main menu). Now the lists look integral to the window, not something just dropped there.

The remainder of the aesthetics are up to you.

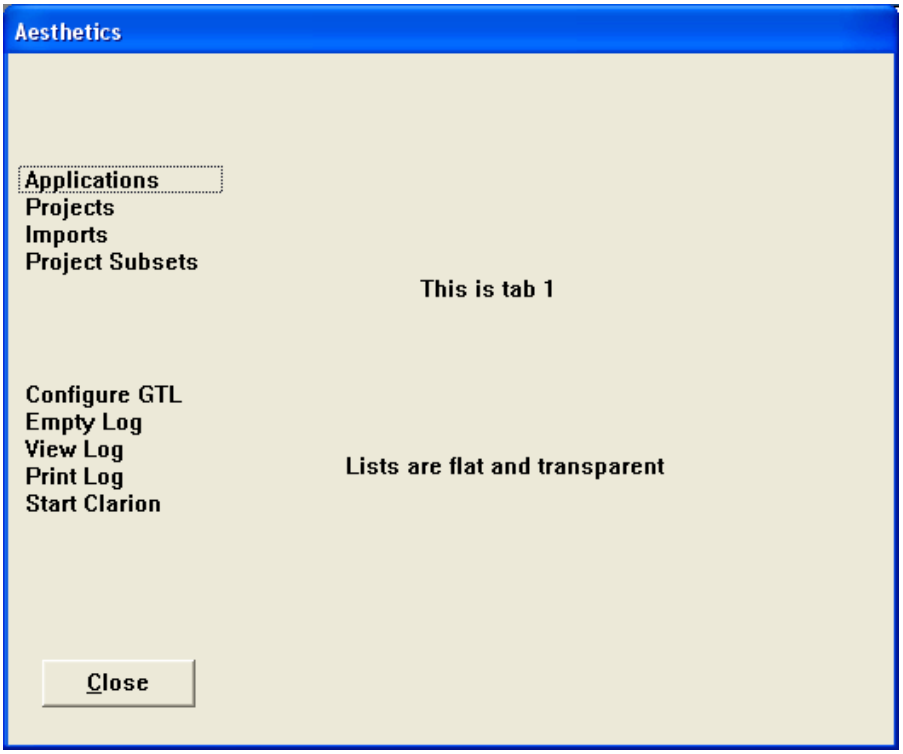


Figure 3. Flat/transparent list controls

Expanding Menus

Expanding menus, similar to those in Outlook/Outlook Express, have long been desired by Clarion programmers. They are now easy. Look at OE's menus in Figure 4.

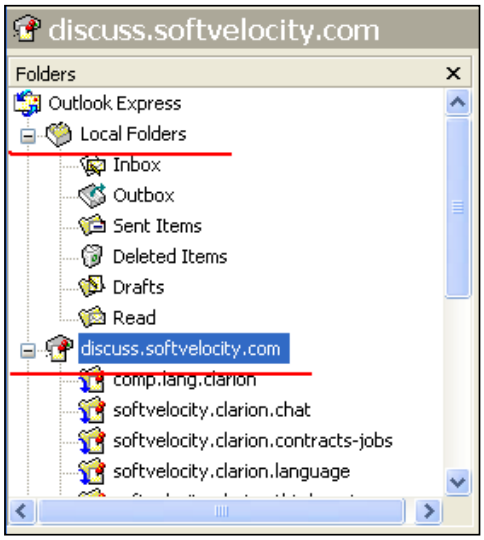


Figure 4. Outlook Express menu

It looks like a tree control, doesn't it?

A Clarion Relation Tree requires a file (at least one) or you may create a tree using Clarion queues directly (see James Cooke's articles on handcoding trees). However, a tree remains a list. Clicking on a list allows checking Choice(?TreeControl) and that means that I can, once again, act on the return value.

```
Case Choice(?RelTree:2)
Of 2 ! Gen & Make
  Self.GenMake
Of 3 ! Gen Only
  Self.GenOnly
Of 4 ! Export APP
  Self.AppExport
Of 5 ! Export DCT
  Self.DCTExport
End
```

There is one caveat. A tree control has a *header record*, a descriptive line. There may be more than one. If the header is clicked, it will return a value but that value does not correspond to a called procedure or a tab selection or to any action in which I may be interested. The first list record is the header so my code starts at choice 2.

The point here is that you need to be careful in your Case structure with trees.

Also, with trees, instead of checking for the returned pointer value, you could check for the words displayed in the tree. In that case:

```
Get(Queue:RelTree:1,Choice(?RelTree:2))
```

will return the displayed line. Change your Case structure accordingly.

Using all of the techniques discussed above, GTL ends up looking much cleaner:

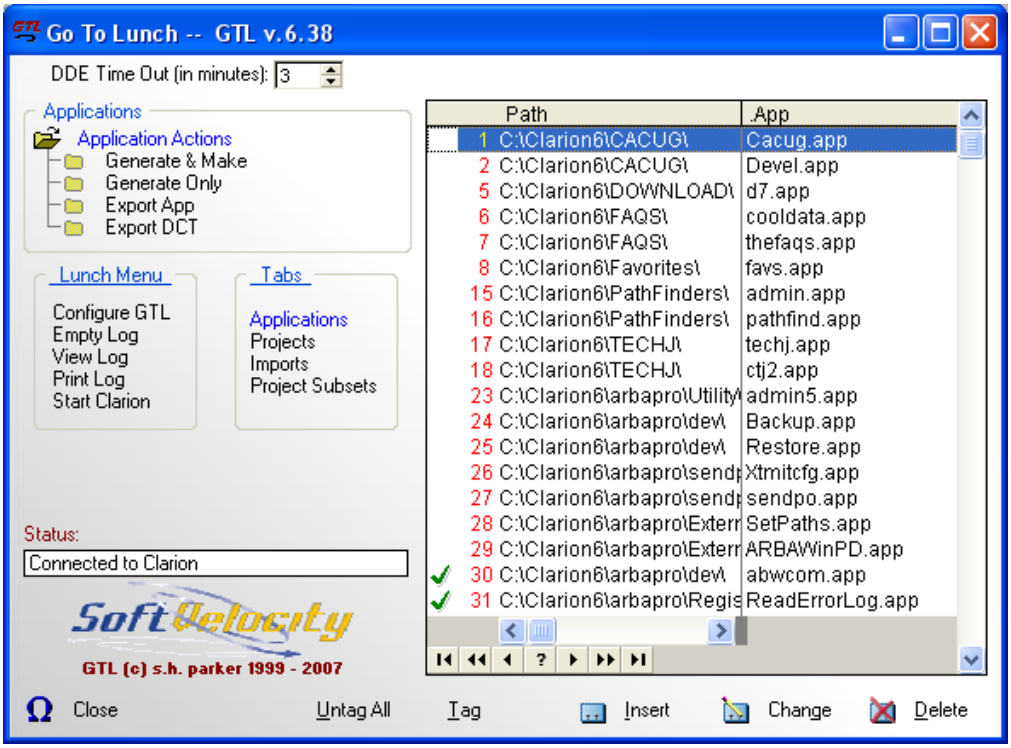


Figure 5. GTL final user interface

Note that in the tree, I only use one file, so there are no plus/minus icons. I could use two files, one to represent each of the three group boxes and a second to hold the individual selections. In that way, I could have a single, expanding/contracting, menu to control everything. In this case, however, that seemed a bit too much.

Oh yes, consider this the formal announcement of the availability of this version of GTL. There are also some new features; see the readme for more information.

Summary

I started worrying about GTL's user interface and writing this article in July 2004. Just in time for C7 and the deprecation of the need for batch compilers, I got it done. By the way, GTL638 compiles very nicely in C7.

Download the source

Download GTL638

Reader Comments

Posted on Friday, January 11, 2008 by Thomas Ruby

Nice UI. Cool new icon too.

Posted on Sunday, January 13, 2008 by Neil Worley

This looks very good. The only issues I have and I guess it is because you have changed the name. The installer still tries to create a gtl62.exe based shortcut. Second the first time I started GTL, it told me I did not have clarion open. When I started the clarion IDE from within GTL it tried to open it four times and only accepted it once.

Neil

Posted on Monday, January 14, 2008 by Steven Parker

Neil, thanks. Shortcuts are fixed and the new install is available for download.

Posted on Wednesday, January 16, 2008 by Carl Barnes

Choice(?Sheet) returns the number of the tab selected i.e. 1,2,3,4. An alternate method is to use ?Sheet{PROP:ChoiceFEQ} which returns the FEQ of the Tab selected. Using this method has the advantage of the CASE code still working correctly if the tabs are reordered, or a new tab is inserted, which would change the CHOICE() numbers.

So instead of:

```
CASE Choice(?Sheet)
OF 1 ; Do TabOneRtn
OF 2 ; Do TabTwoRtn
```

Write it this way:

```
CASE ?Sheet{PROP:ChoiceFEQ}
OF ?Tab:1 ; Do TabOneRtn
OF ?Tab:2 ; Do TabTwoRtn
```

PROP:ChoiceFEQ can also be used to change tabs similar to the way Prop:Selected works, e.g. ?Sheet{PROP:ChoiceFEQ}=?Tab:2.

Posted on Sunday, January 27, 2008 by Edvard Korsbfk

At my place, we are 4 programmers at the time beeing.
GTL is a very important part of my toolbox - Not so with the rest.
Reason - That if a procedure don't behave, the program hangs.
The last change to handling of not compiling app's is a real step forward.
And this program is free...
I never forget how friendly you were at Devcon in 2004 when I came in from Denmark - I can see, that you have not changed.

Add a comment

Clarion Magazine

Using CHOOSE With PROP:SQL

by Steven Parker

Published 2008-01-10

> *The solutions shown here are not mine. Jim Gambon and J. Shankar provided the SQL expertise. The problems, as always (as usual?), were mine.*

Choose is a very powerful addition to the Clarion language, making its debut, I believe with ABC (C4). It's so useful that I wanted to apply it to PROP:SQL queries, only Choose isn't part of the SQL grammar. It turns out there is a way, but first I need to explain what Choose is and does.

Choose "returns the chosen value from a list of possible values." It evaluates the first argument and returns the value from a list I supply. It is well worth your time to look at the Clarion docs on this subject.

Suppose I am doing a sales report and I show the user a filter window. On that window, the user can choose "ascending" or "descending" sorting. That is, the user sees radio buttons with the words "Ascending" and "Descending" but I return one of two strings, 'desc' or 'asc', to the report from this filter window.

Now, I want to update the report title based on the returned value. If the user selected descending order, the report title should be "Top Selling Agents." If ascending is chosen, the title should be "Worst Selling Agents."

To effect this, late in INIT or in OpenReport:

```
Case LOC:Sort
Of 'desc'
  LOC:Title = 'Top Selling Agents'
Of 'asc'
  LOC:Title = 'Worst Selling Agents'
End
```

Or, simply:

```
LOC:Title = Choose(LOC:Sort = 'desc', |
   'Best','Worst') & ' Selling Agents'
```

While this is a trivial example, it does show how much typing Choose can save. Also see Carl Barnes' Using CHOOSE() To Concatenate Data for a non-trivial example.

More significant would be a case where an inventory item has a sale price and a normal price field. Let's say I want to report on the actual selling price.

```
If INV:OnSale = 0
  RPT:SellingPrice = INV:Price
Else
  RPT:SellingPrice = INV:DiscountPrice
End
```

Or, simply:

```
RPT:SellingPrice = |
  Choose(INV:OnSale=0,INV:Price,INV:DiscountPrice)
```

Or, perhaps:

```
RPT:SellingPrice = |
    Choose(INV:DiscountPrice=0,INV:Price,INV:DiscountPrice)
```

Perhaps this is bad design. Perhaps whether or not an item is on sale should be determined by looking it up in a Sale file. If it is found, it is on sale, use that price. If it is not found, it is not on sale, use the base price from inventory.

Perhaps this is a case of under-normalization. But, and this is the important part, it is a circumstance that I thought I had in a database I was working with.

The problem with PROP:SQL

The problem, however, is that my Inventory file was an MS SQL table, not a flat file. And I was trying to use a Prop:SQL statement for my report filter. I thought it might look like this:

```
Select i.PLU, i.PartNumber, i.Description,
  Choose(INV:OnSale=0,INV:Price,INV:DiscountPrice)as Price
  From Inventory i order by PLU
```

Of course, this couldn't work. SQLServer doesn't understand "Choose." I didn't even think of trying this in Management Studio, MS SQL's administrative environment. Neither could I conceive of how to mix SQL with Clarion statement in a Prop:SQL statement inside my .APP.

> *At this point, I begin to ask serious questions about why I don't just do a standard Clarion report. No SQL, just loop through the records and use standard Choose-style code like that shown at the beginning of this article.*
>
> *I have no good answer to this question.*

Jim Gambon to the rescue! Jim pointed out MS SQL's CASE statement (also available in MySQL, Oracle, PostgreSQL, and probably other databases). It is not unlike Clarion's CASE statement and it *can* be used in a SQL query:

```
Select PLU, PartNumber, Description,
  'price' =
    Case
    When OnSale = 0 then
      Price
    else DiscountPrice
    end
 from Inventory
```

MS' documentation states that CASE "evaluates a list of conditions and returns one of multiple possible result expressions." It can take one of two forms:

> The simple CASE function compares an expression to a set of simple expressions to determine the result.
>
> The searched CASE function evaluates a set of Boolean expressions to determine the result.

Further, from MS' docs:

Simple CASE function:

```
CASE input_expression
    WHEN when_expression THEN result_expression
   [ ...n ]
    [
   ELSE else_result_expression
    ]
END
```

Searched CASE function:

```
CASE
    WHEN Boolean_expression THEN result_expression
   [ ...n ]
    [
   ELSE else_result_expression
    ]
END
```

Eureka! The SQL statement, above, works not only in Management Studio, it also works on the right side of:

```
Inventory{PROP:SQL} = ...
```

But Wait! There's more...

No good deed, it is said (and said correctly), goes unpunished. Having "mastered" the first report, I had to do a sales summary report.

The "items sold" table (INVHST) has the item's identifiers, quantity sold, list price, on-sale price and date sold, etc. Each record is a detail for the item for a receipt.

Here, I need to use the sale price, if it exists, otherwise the regular price. Well, thanks to Jim, I know how to do that. But, this time, I need to total quantity sold, number of receipts on which the item appears and total selling price. I need to total the "Case" above.

> *This time, I* do *know why I want to be able to use SQL. A standard Clarion report would have to be a "two pass special." This is just the sort of thing at which SQL is supposed to excel.*

J. Shankar is my knight in shining armor this time.

Shankar points out that MS' Case can be the argument of an aggregation:

```
Select Description, PLU, PartNumber,Sum(QTY) as Quantity,
 Count(Distinct ReceiptNo) as TimesSold
 Sum(Case
  When NetDiscount = 0 then
    Extension
```

```
  else NetDiscount
  end) as SoldPrice
from INVHST
where (SaleDate between <BeginDate> and <EndDate>)
group by PLU, PartNumber, Description
order by Description
```

Summary

Most of my writing has been about my learning curve. Some of it has been on topics that I knew others would find useful. In each case, I have ended with "the lesson I learned."

But, this time, I knew the lesson before I started. That lesson is: select the appropriate news group, state your problem as clearly as you can and, the vast majority of the time, you will get a solution.

I just never cease to wonder at the depth of knowledge and generosity of this community.

Jim and Shankar, thank you.

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

Reader Comments

Posted on Wednesday, January 16, 2008 by Wayne Freeman

Hi Steve, good article.

Couldn't you have also used Choose() while building the string containing the query that Prop:SQL would pass?

Of course, then you wouldn't have had such a wonderful opportunity to learn some more SQL!

.....

Posted on Wednesday, January 16, 2008 by Steven Parker

That would have been much too ... desireable.

Actually, I need the contents of one column to determine which of the others I needed to get.

Now, I suspect, if I didn't want to force feed myself SQL, I could have returned all three and handled it in the report.

Dang! Why didn't I think of that then!

Add a comment

Clarion Magazine

The ClarionMag Blog

Get automatic notification of new items! RSS feeds are available for:

XML All blog entries
XML All new items, including blogs

Blog Categories

- »All Blog Entries
- »Clarion 7 Clarion.NET
- »Future Articles
- »News flashes
- »Nifty Stuff

Let's see your pics!

Direct link

Posted Monday, January 28, 2008 by Dave Harms

Photographer to the (Clarion) stars Leroy Schulz is again running the Andrew's Kitchen photo showcase. Andrew's Kitchen, if you don't already know, is one of several private newsgroups available to Clarion Magazine subscribers (check your subscription email or contact me for access). You don't, however, have to be a subscriber to participate in the photo event.

Here's the blurb:

> Back for its third year, The Best of Andrew's Kitchen 2007 is meant as a showcase of a world inhabited by Clarion programmers. Join other friendly Clarion developers in showing off what has been beautiful, unique, humorous, touching, memorable, or just plain meaningful in your 2007.
>
> Details:
>
> 1. Email photos to mail@frostbytes.ca. *One photo per email please!*
> 2. Please submit images as high-resolution JPG. (Images will be down-sampled as necessary.
> 3. Photos must have been shot from January 1, 2007 to December 31, 2007.
> 4. Include the following information in the body of your email:
>   A. Title of image.
>   B. Photographer name.
>   C. Location.
>   D. Date.
>   E. Short (2-3 sentence) description.
>   F. Technical data. (Optional.)
> 5. Five entries per person maximum.
> 6. Entries must be received by February 8, 2008.
> 7. Copyright remains with the individual photographer, but will be made visible on the website and/or a downloadable slide show.

View this year's entries at http://www.frostbytes.ca/gallery/bestofak2007.

View the 2006 entries at http://www.frostbytes.ca/gallery/bestofak2006.

View the 2005 entries at http://www.frostbytes.ca/gallery/bestofak2005.

Clarion# app on a Mac

Direct link

Posted Monday, January 14, 2008 by Dave Harms

Wayne Freeman has reported success in running a simple (i.e. "Hello world") Clarion# application as a Mono app on a Mac, under X11. Mono is a Novell-sponsored open source project which lets you develop and run .NET client and server applications on Linux, Solaris, Mac OS X, Windows, and Unix.

I've been trying to get a web app running on a Linux box, using Mono's XSP web server for ASP.NET. In ASP.NET applications, source is compiled by the web server; while it's possible to tell XSP to use the Clarion# language, the Clarion# CodeDOM provider isn't yet set up to handle Unix-style paths with / instead of \, so compilation fails. But this appears to be a relatively easy fix, and SV is aware of the problem.

Nine years and counting...

Direct link

Posted Friday, January 11, 2008 by Dave Harms

Welcome to 2008! Unoffically, this is the start of Clarion Magazine's 10th year of publication. Officially, the ninth anniversary of the very first ClarionMag article is this February 8th. Time flies when you're publishing!

Last month was Clarion Magazine's first ever Clarion.NET/Clarion# feature issue. It's hard to really do any kind of justice to a topic as complex as .NET development in just one issue, particularly since Clarion# developers have the opportunity to write not just desktop, but web and mobile apps as well. It's all quite distracting: as I explore Clarion development on .NET I find myself frequently sidetracked by interesting possibilities, such as writing GPS-aware apps for my Motorola Q9h mobile phone or implementing bookmarking XML schemas for Clarion Magazine.

As interesting as Clarion# is, even without an AppGen, C6 is still where most Clarion developers earn their coin, and this month the focus is squarely back on the current product, where it will stay for some time. But you can expect to see continuing coverage on the .NET side of things. In particular I've been doing some web development work with Clarion#, and if that continues to go well I should have a new series up in the near future.

I wish you all a happy, healthy, and prosperous New Year.

Dave Harms, Publisher
