Clarion Magazine

Clarion News

- » New DMC Video
- » Lindersoft Vacation Schedule
- » SetupBuilder 6.8 June 2008 Update
- » iQ-Notes 3.64
- » SimGlobalButtons Template Enhanced
- » Clarion Developer Hosting Sale
- » Ready-to-go Logos $49
- » BuildAutomator Goes Gold
- » Clarion Version Switcher, Ver 2
- » CapeSoft Price Increases June 17th
- » CapeSoft Releases TabTree
- » NeatMessage 2.15
- » J-Zip 1.01
- » J-Skype Public Chat
- » J-Zip Released
- » SealSoft Discount Ends June 10
- » Clarion Desktop 4.11
- » J-Skype Week
- » SetupBuilder 6.8 May 2008 Hotfix
- » More Wallpapers
- » Unofficial Aussie DevCon Wallpaper

[More news]

- » Clarion.NET FAQ
- » Clarion# Language Comparison
- » Generic Types In Clarion# - Advanced Techniques
- » Generic Types In Clarion#

[More Clarion & .NET]

[More Clarion 101]

Latest Free Content

- » Video of New Application Generator
- » DevCon/Server Info On The Blog
- » Source Code Library 2008.05.31 Available

[More free articles]

Clarion Sites

- » Mitten Software

Clarion Blogs

Latest Subscriber Content

Aussie DevCon 2008: Geoff's Thoughts

Geoff Spillane, organizer of the 2008 Aussie DevCon, reflects on the conference.

Posted Monday, June 30, 2008

SQL Processing and Class Interfaces

John de la Torre shows how to use Clarion interfaces to advantage when running generic SQL processes and displaying a result. John's code also illustrates the use of virtual tables.

Posted Monday, June 30, 2008

Single Sign-on Using ActiveDirectory

Microsoft's Active Directory provides a variety of services, including authentication. And sometimes clients who use Active Directory for authentication prefer to use that system rather than your application's custom login. Nardus Swanevelder shows how to integrate Active Directory credentials with your existing security system.

Posted Friday, June 27, 2008

Running Clarion On Vista 64

The old Clarion IDE is a 16 bit application and won't run natively on a 64 bit operating system. But you can run it in a 32 bit virtual machine. Mark Riffey tries out Microsoft Virtual PC and VMWare Workstation.

Posted Thursday, June 26, 2008

Aussie DevCon Day 4 Training

In Day 4 of the Aussie DevCon training sessions Bob Zaunere showed how to develop Compact Framework (mobile) aplications with Clarion#. Geoff Robinson reports.

Posted Wednesday, June 25, 2008

Boost Compile Times By Putting The ABC Classes In Their Own DLL

Have you ever noticed now much time the IDE spends compiling the same ABC classes over and over again? The good news is you don't have to waste that time if you don't want to. Just isolate the ABC classes into their own library. But there's a trick, as Benjamin Dell explains.

Posted Wednesday, June 18, 2008

Creating Directories With #SERVICE

The Clarion template language provides a mechanism for calling DOS utlities such as MKDIR. But if your templates need to create directories the undocumented #SERVICE statement is more reliable.

Posted Wednesday, June 18, 2008

Video of New Application Generator

Bob Z has posted a video showing the new Application Generator. Features demonstrated include automatic conversion of applications and dictionaries from C6 to C7, the Applications pad, generating apps without loading the application, the file schema pad updating ad the procedure changes, the window designer, the new formula editor, multiple instances of the IDE open at once, multi-app solutions, generating all apps in a solution, opening multiple applications at once in a single instance of the IDE (very cool!). Overall the AppGen interface is familiar but somewhat flattened via the use of tabs.

Posted Saturday, June 14, 2008

Generic Types In Clarion# - Advanced Techniques

Generic types are sometimes more useful when you know just a little bit about the types you'll be using. The solution: the generic type WHERE clause.

Posted Tuesday, June 10, 2008

Generic Types In Clarion#

Clarion# has a number of new language features, one of which is support for generic types, or generics. Most commonly used for managing lists of objects where the type is not known at compile time, generics are a powerful addition to your programmer's toolbox.

Posted Tuesday, June 10, 2008

DevCon/Server Info On The Blog

There have been a number of postings on the ClarionMag blog recently, thanks to the Aussie DevCon and the Houston data center explosion. Check it out.

Posted Thursday, June 05, 2008

Source Code Library 2008.05.31 Available

The Clarion Magazine Source Code Library has been updated to include the latest source. Source code subscribers can download the May 2008 update from the My ClarionMag page. If you're on Vista please run Lindersoft's Clarion detection patch first.

Posted Wednesday, June 04, 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 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

gNotes 3.1

A re-branded version of gNotes 3.1 has been released. This version comes with precompiled DLL / LIB for C63 9054-9056, partially re-written template code, new installer and more. No black boxes - source code is included. Price for gNotes 3.1 Source Code is USD 79.00. Until July 15 2008 an introductory price of USD 39.00 is available to users of any of the following: gCal, gNotes, gReg, gCalc, gFileFind, gQ, gSec, and LGP. Until July 15th, owners of any above mentioned utilites can purchase gCal 3.11 Source Code at USD 49.00. New customers may purchase one product at normal price and then order next one at special price.

Posted Wednesday, July 02, 2008

DMC Viewer Licensing Changes

DMC Viewer licensing has changed from one copy to unlimited copies (all registered users will receive an updated activation license). Some prices have also been updated to reflect thischange.

Posted Wednesday, July 02, 2008

DMC Runtime Version

DMC can now be called with command line arguments. DMC closes automatically at the end of processing. All tasks are logged along with errorcodes. This feature will be included in DMC from the next version on.

Posted Wednesday, July 02, 2008

Build Automator 1.30.150

Icetips Creative, Inc. has released Build Automator version 1.30.150. If you are a licensed user, you can use the "Help | Check for updates" from the main menu in the Build Automator to download and update the software automatically. You can purchase the Build Automator with a 60 day Maintenance Plan for US$ 99.00 or you can purchase it with a 1 year Maintenance Plan for US$ 149.00 Icetips Creative, Inc. has recently set up a partnership with Indigo Rose, makers of Setup Factory and MSI Factory and will start developing actions for those products in July. Icetips has also entered a technological partnership with CodeGear, makers of Delphi, RAD Studio and C++ Builder, now being acquired from Borland by Embarcadero Technologies.

Posted Wednesday, July 02, 2008

SetupBuilder 6.8 Build 2266 Developer Edition Hotfox

A downloadable hotfix for SetupBuilder 6.8 Build 2266 (Developer Edition) is available.

Posted Wednesday, July 02, 2008

Aussie DevCon Pics

Stu Andrews has posted some pics from the 2008 Aussie Devcon.

Posted Wednesday, July 02, 2008

Aussie DevCon Comments

Read comments from DevCon participants at Geoff Spillane's message board.

Posted Wednesday, July 02, 2008

DMC 1.4.0.1

DMC 1.4.0.1 is now available. Changes include: SQL query environment; Clone to DAT/DBF; User option for a Sum Up display screen of the Main Wizard (with canceling possibility); Direct access in Viewer to the last 25 most recently used tables (all formats); All Data Management screens redesigned to accommodate more space (better visibility); Data Management Task wizard - define step by step your task (change source and / or destination settings without losing pace); Create ODBC wizard; User option to read only first 50 records before reaching the Mapping tab; Define the Color of all List Box records; List of all SQL Key's in Structure Tab - Structures stored in clipboard; List by ... processes changed to working on queues (speed); Processing of Columns to define if DATETIMEs are present also changed to working on a queue (speed); Bug fixes.

Posted Wednesday, July 02, 2008

Function Points Strategy for Business Information System Estimating

A new Whitemarsh Information Systems short paper on function point analysis is now available for download. See item #13.

Posted Wednesday, July 02, 2008

New DMC Video

A preview video of the new DMC veatures is now available. This is a 15MB download.

Posted Friday, June 20, 2008

Lindersoft Vacation Schedule

The Lindersoft office will be closed for vacation from Saturday, June 28 to Sunday, July 6. During this time the online store will remain open. Orders for new licenses will be processed immediately. Upgrade orders and subscription extension orders will be processed when the office reopens. Emergency support will be provided on a limited basis.

Posted Friday, June 20, 2008

SetupBuilder 6.8 June 2008 Update

The SetupBuilder 6.8 June 2008 Update is now available. This release is free of charge to all SetupBuilder customers who have an active SetupBuilder maintenance subscription plan. If you do not have an active subscription plan, please contact your account manager at sales at lindersoft.com.

Posted Friday, June 20, 2008

iQ-Notes 3.64

New in iQ-Notes 3.64: Play a generic Open/Close/Delete/Alarm sounds when opening, closing and deleting notes; Set non-active notes to a transparent background; Switch to confirm when deleting notes; More flexibility and options for Find features including drop-downs of history; Title can automatically be created when you start typing the details of a note; Integrated HTML help and better layout of User Options tab; Preview contents of notes while in Manage Notes detail screen; New FTP socket tool with better synchronization with Vista.

Posted Friday, June 20, 2008

SimGlobalButtons Template Enhanced

The SimGlobalButtons Template now let you exclude procedures from the automatically setting of button icons and options throughout an application. The upgrade to version 1.01 is free to registered users. The template can be purchased for $29.50 US from ClarionShop.

Posted Friday, June 20, 2008

Clarion Developer Hosting Sale

Now through the end of June Oak Park is running a sale on its shared hosting plans, starting as low as $3.99 per month for Clarion Developers and their clients.

Posted Friday, June 20, 2008

Ready-to-go Logos $49

1st Logo Design offers a selection of ready-to-go logos at $49. These logos are only sold once to retain exclusivity, once sold they are removed from the available list and never sold again. Logos delivered in 300 dpi B&W AI and CDR (vector logos only), PSD, TIFF, PDF, PNG, 72dpi PNG, BMP, JPG, GIF in 800px, 700px, 600px, 500px, 400px, 300px, 200px, 150px and 100px widths.

Posted Friday, June 20, 2008

BuildAutomator Goes Gold

Icetips Creative has released the gold version (1.30.100) of Build Automator. For the final release the Standard Edition has been dropped and the price of the Developer Edition has been lowered to US$149 with a one year Maintenance Plan or US $99 including a 60 day Maintenance Plan. The Maintenance plan is required in order to install updates to the software. Additional features will be available as plugins.

Posted Friday, June 20, 2008

Clarion Version Switcher, Ver 2

Dirt simple application to manage multiple Clarion versions on the same machine. Works with any version of Clarion including multiple versions of Clarion 7 and Clarion.NET. Full source provided, APP file created with C6.1.9033. v2 adds comment, tool tips and rapid selection across multiple versions.

Posted Monday, June 09, 2008

CapeSoft Price Increases June 17th

A number of CapeSoft products will go up in price on June 17th, including: AnyFont (from $49 to $69); MessageBox (from $59 to $69); NetTalk 4 Upgrade (from $199 to $249); NetTalk 4 (from $399 to $499); RightReports (from $249 to $349) (still in beta); Tearoff (from $49 to $69).

Posted Thursday, June 05, 2008

CapeSoft Releases TabTree

TabTree allows you to add a Sheet-Management control to a window. It creates a list of all the tabs in a list box. You can then navigate between tabs using either the keyboard or mouse or the list itself. TabTree is available for an introductory special of $69 until June 17, 2008, when the price goes up to $99. Users upgrading from SimTabTree can purchase an upgrade for $49 until June 17, 2008, when the price will go up to $69.

Posted Thursday, June 05, 2008

NeatMessage 2.15

NeatMessage 2.15 includes an urgent fix to a problem with the Enter key not always firing the selected button. NeatMessage is a Message Box replacement from Huenuleufu Development.

Posted Thursday, June 05, 2008

J-Zip 1.01

J-Zip 1.01 adds an optional progress window.

Posted Thursday, June 05, 2008

J-Skype Public Chat

The new public chat room feature in J-Skype 2.5 lets you participate in Skype chats from within a Clarion application.

Posted Thursday, June 05, 2008

J-Zip Released

Strategy Online has released J-Zip for Clarion. This is a template wrapper around the 7Zip library. For now there is a single code template to add files to a ZIP archive, and another code template to extract files. Demo available; price is $15.

Posted Thursday, June 05, 2008

SealSoft Discount Ends June 10

Anyone purchasing a SealSoft product at ClarionShop by June 10 can receive a 40% discount. Include 'SealSoft DevCon08 Discount' in the comment field. Buyers at Motleysoft.com will also receive the discount.

Posted Thursday, June 05, 2008

Clarion Desktop 4.11

Clarion Desktop 4.11 is now available.

Posted Thursday, June 05, 2008

J-Skype Week

The gold release of J-Skype 2.50 is due out this Friday.

Posted Thursday, June 05, 2008

SetupBuilder 6.8 May 2008 Hotfix

Lindersoft has released a SetupBuilder 6.8 hotfix which addresses a command line compiler issue. Web Update can be accessed via "Help | Check for Updates..." in your SetupBuilder IDE.

Posted Thursday, June 05, 2008

More Wallpapers

Jesus Moreno has posted some additional free wallpapers.

Posted Thursday, June 05, 2008

Unofficial Aussie DevCon Wallpaper

Jesus Moreno has an unofficial Aussie DevCon wallpaper available for download.

Posted Thursday, June 05, 2008

Clarion Magazine

Generic Types In Clarion# - Advanced Techniques

by Dave Harms

Published 2008-06-10

This article is the fourth in a series on test drive development and generic types. In fact the first two articles weren't about generic types at all but instead laid the groundwork for a test-driven development (TDD) environment using NUnit. These articles also introduced the concept of a SortOrder class and another SortOrderList class to manage a list of sort orders for a browse. As you might recall I created those classes to support a particular kind of web application (which itself will be the subject of future articles).

In the third article I added the concept of a DataFilter class to support browse filtering. I evolved my SortOrderList class first into a non-generic ObjectList class and then into a generic ItemList class which used one of the .NET generic collections classes to store the actual list of objects.

My generic ItemList class lets me store objects of any type, not just DataFilter and SortOrder objects. But that has a downside as well. Because ItemList doesn't know anything ahead of time about the types of objects it will be dealing with it can't treat those objects as anything more complex than instances of Object (at least not without using reflection). And there are several situations where it would be handy to be able to call methods on DataFilter and SortOrder. I'll solve this problem by employing a WHERE clause on my generic type.

Objectives

I have two objectives I want to achieve. The first is to clean up what I see as some duplicated code in my design, and the second is to find a way of getting a list of ids and descriptions out of ItemList. The code cleanup is optional but useful; the list of ids and descriptions is essential. More on that in a moment.

Here's a quick refresher on the concept behind the classes I'll be discussing in this article. I'm developing some web applications which include pages that roughly correspond to browses and forms. Browses, whether web or desktop, have some common features including sorting and filtering. In my case I sometimes want to present drop-down lists to the users to let them change sort orders and/or filters. In HTML I do this with a SELECT statement:

```
<select name="SortOrderID" >
   <option selected value="id">User ID</option>
   <option value="firstlastname">First/Last name</option>
   <option value="lastfirstname">Last/First name</option>
</select>
```

As this HTML snippet shows I need at least two items of information, an id string for each option and a description for each option (actually I could get away with using the description as the id but I prefer to keep these separate).

That takes care of the client side; on the server side I need some additional sort order information, which for purposes of illustration I've reduced to just two fields: one for the field name and one for an ascending/descending flag (the real-world implementation of this class is somewhat more complex).

I use a similar approach for lists of filters, except that on the server side I keep a string field containing an SQL WHERE clause. (For more on the client/server interaction aspect please see TDD Part 1.)

Here's the core source code as I left it at the end of the last article. The I have a DataFilter class, a SortOrder class and a ListItem class which can manage a list of either DataFilters or SortOrders, depending on how I implement ListItem:

```
DataFilter          class,public
id                  string
description         string
whereclause         string
Construct           procedure(string id,string description,|
                      string whereclause)
                    end

DataFilter.Construct        procedure(string id,string description,|
                              string whereclause)
  code
  self.id = id
  self.description = description
  self.whereclause = whereclause



SortOrder           class,public
id                  string
description         string
column              string
ascending           bool
Construct           procedure(string id,string description,|
                      string column,bool ascending)
                    end

SortOrder.Construct     procedure(string id,string description|
                          ,string column,bool ascending)
  code
  self.id = id
  self.description = description
  self.column = column
  self.ascending = ascending



ItemList            class<T>,public
```

```
Items                Dictionary<string,T>
DefaultID              string
CurrentID              string,private
Add                  procedure(string id,T)
Construct              procedure
Get                  procedure,T
Get                  procedure(string id),T
                 end

ItemList<T>.construct      procedure
  code
  self.Items = new Dictionary<string,T>()



ItemList<T>.add    procedure(string id,T item)
  code
  if self.Items.Count = 0
    self.DefaultID = id
  end
  self.Items.Add(id,item)



ItemList<T>.Get    procedure
  code
  if ~self.CurrentID = null
    return self.Get(self.CurrentID)
  else
    return self.Get(self.DefaultID)
  end

ItemList<T>.Get    procedure(String id)
  code
  if id = null or ~self.Items.ContainsKey(id)
    throw new ItemNotFoundException('No item found for id ' & id)
  else
    self.CurrentID = id
    return self.Items[id]
  end
```

To keep things simple I've left out the ToString methods in DataFilter and SortOrder.

Getting a list of IDs and descriptions

I explained earlier that for each list of DataFilters or SortOrders I want to present to the user I need a corresponding list of ids and descriptions. I'll leave aside the specific mechanism by which these lists are converted into HTML (if you absolutely must know, I'm using the NVelocity view engine with Monorail); the salient bit is that a simple array of objects, each containing an id and description, will do the job nicely. And the logical place to get that list is from ItemList, right?

Unfortunately, although ItemList can be told what kind of object (DataFilter or SortOrder) it will use in place of T, it gets this information at runtime, not compile time. At compile time T is really just Object, the root class in the .NET class hierarchy. If I attempt to treat T as if it is DataFilter and I try retrieve the id and description properties I'll get compiler errors telling me that id and description are not members of T. I could use reflection to tease information out of the T object at runtime but that's a lot of work and there is a better way.

What I really need is a way to tell ItemList that T is not just Object but something that is guaranteed to have certain properties and/or methods. And that means I'll need either an interface that's implemented by both DataFilter and SortOrder or a base class from which DataFilter and SortOrder are derived.

Base class or interface?

An interface is basically a contract that says a class will implement certain methods. A class that implements an interface must contain the source for those methods since the interface cannot contain any data or source code of its own. In this situation an interface is probably not the best option since I would end up duplicating the storage of the id and description strings.

A base class can contain whatever data and code (say, accessor methods) are common to all derived classes, so there's no need to duplicate anything. The main downside to inheritance is that Clarion#, like Clarion, only allows single inheritance. You can't create a class that's derived from more than one other class. In this case I don't foresee single inheritance causing any problems.

Here's my base class for DataFilter and SourceOrder. I've called it WebPageItem and its main purpose is to contain the information I'll need when building SELECT statements on the web page:

```
WebPageItem         class,public
id                  string
description         string
Construct           procedure
Construct           procedure(string id,string description)
                    end

WebPageItem.Construct       procedure
  code

WebPageItem.Construct       procedure(string id,string description)
  code
  self.id = id
  self.description = description
```

Now I can change my SortOrder and DataFilter classes to use this base class. To differentiate these versions I've called them SortOrderWPI and DataFilterWPI:

```
SortOrderWPI        class(WebPageItem),public
column              string
ascending           bool
Construct           procedure(string id,|
                      string description,string column,|
                      bool ascending)
                  end

SortOrderWPI.Construct      procedure(string id,|
                              string description,string column,|
                              bool ascending)
  code
  self.id = id
  self.description = description
  self.column = column
  self.ascending = ascending

DataFilterWPI       class(WebPageItem),public
whereclause         string
Construct           procedure(string id,|
                      string description,string whereclause)
                  end

DataFilterWPI.Construct     procedure(string id,|
                              string description,string whereclause)
  code
  self.id = id
  self.description = description
  self.whereclause = whereclause
```

Once again I've left off the ToString methods for brevity. As you can see SortOrderWPI and DataFilterWPI add their unique properties and have appropriate constructors.

Next I create a new version of ItemList called WebPageItemList and I tell it to only accept types that are, or are derived from, WebPageItem:

```
WebPageItemList     class<T> where(T=WebPageItem),public
```

Now anywhere I have T in my class I can assume I can use WebPageItem's methods and properties. And that brings up another issue that bugs me about the previous ItemList implementation.

Cleaning up the code

If you go back to the test code for ItemList you'll see this method:

```
TestItemList.CreateSortOrderTestData   procedure
x                                      long
   code
   self.SortOrderList = new ItemList<SortOrder>()
   loop x = 1 to 10
      self.SortOrderList.add('id' & x,self.CreateSortOrder('id' & x))
   end
```

The CreateSortOrder method creates the actual SortOrder object which is added to SortOrderList, but SortOrderList.Add still expects the id to be passed separately even though that id and the id contained inside the SortOrder object are identical. That's because ItemList doesn't have any way of extracting the id from SortOrder.

But WebPageItemList knows its dealing with WebPageItem (or a derived class) and it can extract the id (and the description). So I can now clean the add method up and simply pass in the object:

```
WebPageItemList<T>.add      procedure(T item)
   code
   if self.Items.Count = 0
      self.DefaultID = item.id
   end
   self.Items.Add(item.id,item)
```

That's much nicer.

The other thing I can now is add a method to retrieve an array of WebPageItem objects from WebPageItemList. Here's what my test code looks like:

```
TestWebPageItemList.GetWPIArray     procedure
expected                            SortOrderWPI
results                             WebPageItem[]
   code
   self.CreateSortOrderTestData()
   expected = self.CreateSortOrder('id2')
   results = self.SortOrderList.GetWebPageItems()
   NUnit.Framework.Assert.AreEqual(expected.id,results[1].id,'Test 1')
   NUnit.Framework.Assert.AreEqual(10,results.Length,'Test 2')
```

First I create the test data (SortOrder objects, in this case) and then I create a separate SortOrder corresponding to the one I'm looking for. I want SortOrderList (which is declared as WebPageItemList<SortOrder>) to return an array of WebPageItems corresponding to its list of SortOrders. I've created 10 test items as 'id1' to 'id10', but as Clarion# arrays are zero-based the WebPageItem array element that corresponds to 'id2' is actually index 1.

If the id values from both objects are the same then the first test passes. A second tests ensures that all 10 test items are returned in the array.

Here's the complete listing for WebPageItemList - you can see the GetWebPageItems method at the end:

```
   using System
   using System.Collections.Generic



WebPageItemList     class<T> where(T=WebPageItem) ,public
Items               Dictionary<string,T>
DefaultID           string
CurrentID           string,private
Add                 procedure(T)
Construct           procedure
Get                 procedure,T
Get                 procedure(string id),T
GetWebPageItems     procedure,WebPageItem[]
                    end

WebPageItemList<T>.construct        procedure
   code
   self.Items = new Dictionary<string,T>()



WebPageItemList<T>.add      procedure(T item)
   code
   if self.Items.Count = 0
      self.DefaultID = item.id
   end
   self.Items.Add(item.id,item)



WebPageItemList<T>.Get      procedure
   code
   if ~self.CurrentID = null
      return self.Get(self.CurrentID)
   else
      return self.Get(self.DefaultID)
   end

WebPageItemList<T>.Get      procedure(String id)
   code
   if id = null or ~self.Items.ContainsKey(id)
      throw new ItemNotFoundException('No item found for id ' & id)
```

```
    else
        self.CurrentID = id
        return self.Items[id]
    end



WebPageItemList<T>.GetWebPageItems  procedure
wpiArray                            WebPageItem[]
pair                                KeyValuePair<string, T>
wpi                                 WebPageItem

x                                   long
    code
    if self.Items.Count > 0
        wpiArray = new WebPageItem[self.Items.Count]
        x = 0
        foreach pair in self.Items
            wpi = pair.Value ! tryas WebPageItem
            wpiArray[x] = new WebPageItem(wpi.id,wpi.description)
            x += 1
        end
        return wpiArray
    else
        throw new ItemNotFoundException(|
          'There are no WebPageItems in this Items')
    end
```

WebPageItemList uses a generic Dictionary to store the list of WebPageItem (or derived) objects, and consequently the GetWebPageItems method has to use some special code to extract the objects from that collection. First, GetWebPageItems creates a new array of WebPageItem objects; next, it extracts the stored objects (which are most likely of type SortOrder or DataFilter). Each item in the Dictionary is represented by a KeyValuePair<TKey,TValue> generic class, here declared as pair. The foreach statement extracts the key/value pairs into pair, and then the value is stored in wpi. A new WebPageItem is created based on the data in wpi.

Once all the dictionary items have been processed the code returns the new array.

Summary

As useful as generics are, sometimes you need just a little bit more information about the classes used in place of the generic type. The generic type WHERE clause lets you specify a base class from which generic types must be derived or an interface which generic types must implement. With a known set of methods and/or properties in place you can reap the benefits of generic types while writing code that is specific to that base class or interface.

Download the source

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

Posted on Thursday, June 12, 2008 by Stephen Ryan

some basics of generics

http://www.madeincode.com/Articles/CSharp/Generics/Generics.aspx

Add a comment

Clarion Magazine

Generic Types In Clarion#

by Dave Harms

Published 2008-06-10

There are quite a few interesting new features in Clarion#, and one of these is something called generic types, or generics.

Generics are a way of adding increased flexibility to your applications by letting the type of an object be determined at runtime rather than at compile time. They are most commonly used in situations where you need to manage lists of objects and you don't know the object's type ahead of time, but they have a wide range of application.

Based on that short description you may not think that generics are any big deal, but if you've ever written your own Clarion classes there's a decent chance you've been in a situation where you really could have used generic types. So read on; you may find generics a whole lot more interesting than you think.

The setup

This article on generics builds on a previous pair of articles: Agile, Test-Drive Development In Clarion#, Part 1 and Agile, Test-Drive Development In Clarion#, Part 2. In Part 1 I introduced the concept of test-driven development (TDD); in Part 2 I showed how to use TDD to create a library which I use in certain .NET web applications. The key aspect of this web app library is that it manages lists of sort orders (including the ability to track default and previously set sort orders), but it does so using standard, non-generic techniques.

If you haven't read those two articles on TDD, I suggest you do so now, in particular the second article as it presents the SortOrder and SortOrderList classes. By way of review, SortOrder contains the information needed to display a user-selectable sort order on the web page, as well as the information needed to create that sort order on the back end. SortOrderList is a class that simply manages a list of SortOrder objects. (For more details on the overall design see TDD Part 2.)

In this article I'll begin expanding the functionality of that list management class, eventually ending up with an implementation that demonstrates generics in Clarion#.

Adding a filter

Sort orders are a common feature of browses, whether these browses are in desktop applications or in web applications. Filters are another important aspect of browses and it's sometimes necessary for users to be able to choose between different filters just as they choose different sort orders.

It makes sense to me to handle filters much the way I already handle sort orders. My first step is to create a DataFilter class. (And yes, I could use just the name Filter but that causes a conflict with Clarion#'s own FILTER statement and forces me to use the fully qualified class name, which entails a whole lot more typing.)

In my ClarionMag.Articles.Generics project (see TDD Part 2 for further information on this project) I create a new member module and call it DataFilter.cln. I enter the following code below the MAP statement:

```
using System
using System.Text
```

```
DataFilter          class,public
id                  string
description             string
whereclause             string
Construct               procedure(string id,string description,|
                      string whereclause)
ToString                procedure,string,derived
                end

DataFilter.Construct        procedure(string id,string description,|
                          string whereclause)
    code
    self.id = id
    self.description = description
    self.whereclause = whereclause

DataFilter.ToString procedure
sb                  StringBuilder()
    code
    sb.append(self.id & ',')
    sb.append(self.description & ',')
    sb.append(self.whereclause)
    return sb.ToString()
```

The DataFilter class is quite similar to the SortOrder class except that in place of the column and ascending properties it has a whereClause property. And given that the classes are so similar it seems silly to create yet another class called FilterList to manage my list of DataFilter objects. Instead I'd like to make SortOrderList more generic.

Lists of unknown objects

As in the previous article I'm using a test-driven approach. I create a new member file in my ClarionMag.Articles.Generics. Test project called TestObjectList.cln and I add the following code:

```
    using System
    using System.Collections
    using ClarionMag.Articles.Generics
    using NUnit.Framework



    [TestFixture(Description='Non-generic list')]
TestObjectList      class,public
list                ObjectList
CreateFilter            procedure(string id),DataFilter
CreateFilterTestData    procedure
```

```
CreateSortOrder         procedure(string id),SortOrder
CreateSortOrderTestData procedure
                        [Test]
FindFilterByID          procedure,public
                        [Test]
FindSortOrderByID       procedure,public
                    end

TestObjectList.CreateFilter    procedure(string id)
   code
   return new DataFilter(id,id,true)

TestObjectList.CreateFilterTestData      procedure
x                                        long
   code
   self.list = new ObjectList()
   loop x = 1 to 10
      self.list.add('id' & x,self.CreateFilter('id' & x))
   end

TestObjectList.CreateSortOrder   procedure(string id)
   code
   return new SortOrder(id,id,id,true)

TestObjectList.CreateSortOrderTestData   procedure
x                                        long
   code
   self.list = new ObjectList()
   loop x = 1 to 10
      self.list.add('id' & x,self.CreateSortOrder('id' & x))
   end



TestObjectList.FindFilterByID    procedure
expected                           DataFilter
result                           DataFilter
   code
   self.CreateFilterTestData()
   expected = self.CreateFilter('id2')
   result = self.list.Get('id2')
   NUnit.Framework.Assert.AreEqual(expected.ToString(),result.ToString())

TestObjectList.FindSortOrderByID procedure
expected                           SortOrder
result                           SortOrder
```

```
   code
   self.CreateSortOrderTestData()
   expected = self.CreateSortOrder('id2')
   result = self.list.Get('id2')
   NUnit.Framework.Assert.AreEqual(expected.ToString(),result.ToString())
```

I've created two pairs of methods to create test data for DataFilter and SortOrder objects and I've reduced the suite of tests to simply looking for a certain object by its id. Of course the code won't yet compile as I haven't declared my ObjectList class.

Since I have my ClarionMag.Articles.Generics.Test solution open, and that solution also contains the ClarionMag.Articles.Generics project, I don't have to close the current solution. I simply add a new member module to ClarionMag.Articles.Generics and call it ObjectList.cln. I add the following code below the MAP statement:

```
   using System
   using System.Collections



ObjectList      class,public
Objects           SortedList
DefaultID          string
CurrentID          string,private
Add               procedure(string id,Object)
Construct          procedure
Get              procedure,Object
Get              procedure(string id),Object
              end

ObjectList.construct      procedure
   code
   self.Objects = new SortedList()



ObjectList.add     procedure(string id,Object so)
   code
   if self.Objects.Count = 0
      self.DefaultID = id
   end
   self.Objects.Add(id,so)

ObjectList.Get     procedure
   code
   if ~self.CurrentID = null
      return self.Get(self.CurrentID)
   else
      return self.Get(self.DefaultID)
   end
```

```
ObjectList.Get      procedure(String id)
  code
  if id = null or ~self.Objects.ContainsKey(id)
    throw new ItemNotFoundException('No item found for id ' & id)
  else
    self.CurrentID = id
    return self.Objects[id]
  end
```

ObjectList is almost identical to SortOrderList except that it uses Object instead of SortOrder. In .NET (as in many other OO languages) all classes are ultimately derived from a root class called Object. And that means that if I want to handle lists of unknown objects I can simply treat them as instances of Object.

> **NOTE:** Clarion already has a data type called ANY that fills a similar role to Object, at least for simple data types. In fact in Clarion# ANY is a class and like all classes it too is derived from Object, but it adds a bit of functionality for backward compatibility with the Clarion ANY statement.

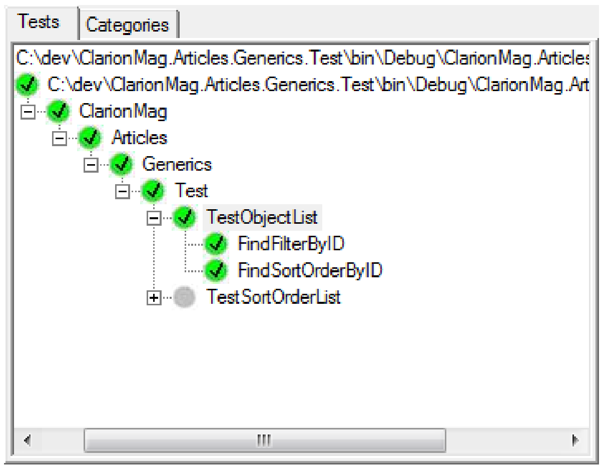

Tests
Categories
C:\dev\ClarionMag.Articles.Generics.Test\bin\Debug\ClarionMag.Articles
C:\dev\ClarionMag.Articles.Generics.Test\bin\Debug\ClarionMag.Art
ClarionMag
Articles
Generics
Test
TestObjectList
FindFilterByID
FindSortOrderByID
TestSortOrderList

Figure 1. Running the TestObjectList unit tests

As Figure 1 shows, my tests run successfully. Problem solved, right?

Not quite. Watch what happens if I make a subtle alternation to the FindSortOrderByID method: I've changed the data type of result from SortOrder to DataFilter.

```
TestObjectList.FindSortOrderByID procedure
expected                    SortOrder
result                    DataFilter ! was SortOrder
  code
  self.CreateSortOrderTestData()
  expected = self.CreateSortOrder('id2')
  result = self.list.Get('id2')
  NUnit.Framework.Assert.AreEqual(expected.ToString(),result.ToString())
```

This code compiles just fine because the ObjectList.Get method returns an Object and DataFilter is just another kind of Object. But when I run this code NUnit reports an "invalid cast" exception, meaning it couldn't convert an object of one type into an object of another type:

> ClarionMag.Articles.Generics.Test.TestObjectList.FindSortOrderByID : System.InvalidCastException : Unable to cast object of type 'ClarionMag.Articles.Generics.SortOrder' to type 'ClarionMag.Articles.Generics.DataFilter'.

The problem is that I've created my test data with SortOrder objects but I'm trying to assign one of those objects to a DataFilter reference. The two type definitions are incompatible and the runtime system cannot perform the cast.

But surely I can create a test scenario to account for this problem! Yes, I can, but that isn't the point. The point is that this is an error that only shows up at *run time*, and ideally I want to catch these kinds of errors at *compile time*. Besides, casting from one type to another takes processor cycles; do it a lot and you could be looking at performance problems.

Wouldn't it be nice to have the flexibility of object references along with compile time type checking? Yes it would, and that's exactly what generic types provide.

Introducing generics

I'm going to present some code a little out of sequence in this section. First I'll show you the test code, as usual. Normally I'd go ahead and write the implementation code next and then run the tests, but in this case I want to demonstrate the test results first before I show the implementation code.

I'll need a new member module in my ClarionMag.Articles.Generics project, and I'll call this one ItemList.cln. Here's the code:

```
  using System
  using System.Collections
  using ClarionMag.Articles.Generics
  using NUnit.Framework



  [TestFixture(Description='Non-generic list')]
TestItemList      class,public
list              ItemList<SortOrder>
CreateSortOrder       procedure(string id),SortOrder
CreateSortOrderTestData procedure
                  [Test]
FindSortOrderByID     procedure,public
                end

TestItemList.CreateSortOrder   procedure(string id)
  code
  return new SortOrder(id,id,id,true)

TestItemList.CreateSortOrderTestData   procedure
x                                 long
  code
  self.list = new ItemList<SortOrder>()
  loop x = 1 to 10
    self.list.add('id' & x,self.CreateSortOrder('id' & x))
  end

TestItemList.FindSortOrderByID procedure
expected                    SortOrder
result                      SortOrder
  code
  self.CreateSortOrderTestData()
  expected = self.CreateSortOrder('id2')
  result = self.list.Get('id2')
  NUnit.Framework.Assert.AreEqual(expected.ToString(),result.ToString())
```

This code assumes I've written a class called ItemList, and for now you'll have to take it on faith that ItemList is able to handle both SortOrder and DataFilter objects. The important thing about ItemListTest is that it's actually telling ItemList which data type to expect via some peculiar new syntax. This code:

```
list              ItemList<SortOrder>
```

declares a reference variable of type ItemList, but it also tells ItemList that it's going to be dealing with objects that are of the type SortOrder. The actual list object is created using similar syntax:

```
 self.list = new ItemList<SortOrder>()
```

I've secretly written my ItemList implementation, so all this code now compiles and the unit test passes. Now watch what happens if I again try to muck about with the FindSortOrderByID method. If I change

```
result                      SortOrder
```

to

```
result                      DataFilter
```

I'll get a compiler error:

```
Variable of type '&ClarionMag.Articles.Generics.DataFilter' can't be reference-assigned a value of type '&ClarionMag.
Articles.Generics.SortOrder' (CLA00341) - c:\dev\ClarionMag.Articles.Generics.Test\TestItemList.cln:76,5
```

In the previous example, using ObjectList, this change compiled fine but resulted in a runtime error. If my testing was incomplete my users would be the ones to find the error, but with generics the compiler sees the flaw in my code. Most importantly I have the flexibility to use my ItemList class with either SortOrder or DataFilter objects; the only difference from ObjectList is that I'm specifying in my code which object type I'm expecting. I can now create two ItemList classes, one for each object type:

```
SOItemList        ItemList<SortOrder>
DFItemList        ItemList<DataFilter>
```

SOItemList will only accept SortOrder objects, and DFItemList will only accept DataFilter objects. And because I'm working with known types I don't incur any of the usual overhead involved in casting one type to another.

How generics work

The idea behind generics is that your code can use *type* information the same way it uses other more traditional kinds of information. The syntax for passing in type information is a bit different from other kinds of information; as you've already seen the new syntax involves the use of angle brackets. Here's the source code for the ItemList class:

```
   using System
   using System.Collections.Generic



ItemList         class<T>,public
Items              Dictionary<string,T>
DefaultID            string
CurrentID            string,private
Add                Dictionary procedure(string id,T)
Construct            procedure
Get                 procedure,T
Get                 procedure(string id),T
               end

ItemList<T>.construct     procedure
   code
   self.Items = new Dictionary<string,T>()



ItemList<T>.add    procedure(string id,T item)
   code
   if self.Items.Count = 0
      self.DefaultID = id
   end
   self.Items.Add(id,item)



ItemList<T>.Get    procedure
   code
   if ~self.CurrentID = null
      return self.Get(self.CurrentID)
   else
      return self.Get(self.DefaultID)
   end

ItemList<T>.Get    procedure(String id)
   code
   if id = null or ~self.Items.ContainsKey(id)
      throw new ItemNotFoundException('No item found for id ' & id)
   else
      self.CurrentID = id
      return self.Items[id]
```

```
    end
```

The first thing to note is that CLASS declaration, which is followed by <T>:

```
  ItemList          class<T>,public
```

The <T> syntax says that anywhere you declare this class you must also include some type information, also enclosed in angle brackets. And since methods are always implemented using the syntax *classname.methodname*, in a generic class you will always see the type information in the CLASS declaration (in this example, <T>) duplicated in the method names. (If you want to add a generic type to the method, that goes after the method name - but that's a subject for another article. For more on generic methods see the Clarion# help under Generics.)

As I showed earlier, TestItemList declares its items property as follows:

```
 list                ItemList<SortOrder>
```

In this case, anywhere you see a T in ItemList you can mentally substitute SortOrder. Take a look at the Add and Get method declarations:

```
 Add                  procedure(string id,T item)
 Get                 procedure,T
 Get                 procedure(string id),T
```

If ItemList is declared as ItemList<SortOrder> the runtime sees these methods like this:

```
 Add                  procedure(string id,SortOrder item)
 Get                 procedure,SortOrder
 Get                 procedure(string id),SortOrder
```

If ItemList is declared as ItemList<DataFilter> the runtime sees these methods like this:

```
 Add                  procedure(string id,DataFilter item)
 Get                 procedure,DataFilter
 Get                 procedure(string id),DataFilter
```

You don't strictly have to use T as the identifier for type information; it's just a convention. This example also uses just one type parameter; you can have multiple type parameters separated by commas. When you have multiple types another convention is to use the following letters in the alphabet, as in:

```
 MyClass          class<T,U,V>,public
```

Sometimes you'll see multiple types declared as T*description* where the T indicates a type and the *description* text differentiates one type from another. It's not that often you'll see a class with more than one or two types, however. Adding a whole bunch of generic types to a class can make it overly complex and difficult to maintain.

Using generic queues

Besides using the type information for the Add and Get methods, ItemList also uses this information to make sure that only the right kind of object is stored in its internal list.

You have two options when it comes to generic lists: generic queues and generic collections. Here's a generic queue to manage my list of items:

```
ItemQueue          Queue<T>,public,type
id                 String
item               T
                   end
```

By now the <T> syntax should be looking familiar. As with the class, you pass the type information when you declare and instantiate the class. Assuming my test code defines ItemList as ItemList<SortOrder> the runtime will substitute SortOrder for T in the previous and following snippets. ItemList is instantiated with the SortOrder type, which means that ItemQueue also gets the SortOrder type and the ItemQueue.Item field is of type SortOrder:

```
ItemList           class<T>,public
Items              ItemQueue<T>
...
                   end



ItemList<T>.construct      procedure
   code
   self.Items = new ItemQueue<T>
```

Using generic collections

Queues aren't your only option; you may also want to use the standard .NET generic collections. Here's an abbreviated listing showing ItemList implementing a generic dictionary:

```
   using System.Collections.Generic




ItemList           class<T>,public
Items              Dictionary<string,T>
...
Construct          procedure
...
                   end

ItemList<T>.construct      procedure
   code
   self.Items = new Dictionary<string,T>()
```

The Dictionary<TKey,TValue> collection class is basically a key-accessible list in which you can define the type of both the key and the value. In this case I know I'm going to always want a string key so I defined the first type as string and the second type as T, which will be defined whenever I create an instance of ItemList. At this time I'm specifying T as SortOrder or ItemList, but as you've probably realized by now I could be using any type at all.

Collections vs queues

Clarion programmers have definitely been spoiled by queues, particularly when it comes to sorting. Queues easily handle multi-field sorting, but that's a more complex task with collections, even generic collections. But for many simple tasks .NET collections are easy to implement and easy to use. If you just need to keep a simple linked list, stack or dictionary of items then .NET collections should probably be your first choice. They generally take less code and unlike queues they don't have to be declared outside the class.

> NOTE: When I began writing this series of articles Clarion#'s support for generic queues was incomplete. Although the bug that completely prevented me from using generic queues has been fixed in build 3421 there's another bug in GET(queue,key) that makes the code a bit ugly. The downloadable source includes a class called ItemListWithQueue which implements a generic queue, and you can substitute that class for ItemList if you wish. All other code discussed in this article uses the .NET generic collections rather than generic queues.

For an implementation such as the one I'm describing in this article I favor the .NET collections, partly because there's less code involved and partly because all of the code is contained inside the class (queues must be declared outside the class, as in C6).

Summary

Generics let you design classes that aren't bound to any one data type, but which take full advantage of compile-time type checking. This means fewer runtime errors, cleaner code and better performance. You can write your own generic classes and queues and you can also take advantage of a number of generic collection classes included with the .NET framework.

The downloadable source contains two solutions, ClarionMag.Articles.Generics and ClarionMag.Articles.Generics.Test. The test solution contains both projects, so you may just want to open that solution. To run the tests you'll also need NUnit, as described TDD Part 1.

Next time I'll discuss advanced uses of generic types.

Download the source

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

Aussie DevCon 2008: Geoff's Thoughts

by Geoff Spillane

Published 2008-06-30

When I put my hand up to organise the Aussie DevCon for 2008 I had just a few simple goals in mind.

- To keep the Clarion community spirit alive and kicking in Australia, as most of the State user groups had all but disappeared.
- To give Soft Velocity an opportunity to show what had been achieved over the last 12 months.
- To give people an opportunity to undertake some quality training, especially in the new products.

To say that I achieved my goals would be grossly understating the reality. No, I'm not blowing my own trumpet because I did little more than provide a vehicle for the real heroes. These heroes all combined brilliantly to deliver a whole week of unforgettable training and presentations.

These heroes took the form of the Soft Velocity team of Bob Foreman, Diego Borojovich and SV President Robert Zaunere. They took the form of the other presenters, namely Bruce Johnson, Michael Summons and Kim Davies. They also took the form of the master of ceremonies, panel host and Saturday night activity manager Stu Andrews. It was their professionalism and obvious preparation that left the lasting impressions. However there was also one more rather intangible component. It was atmosphere, and there was a very good reason for this atmosphere.

As software developers I believe we should all recognise that product development doesn't just consist of two stages, namely unfinished and finished. There's the excitement of concept, the frustration of the building and testing process, the moments of pleasure in discovering new features that can be incorporated, the anticipation of near completion and the final exhilaration of proudly showing the end product.

We all watched and learned as three key players on the Soft Velocity team delivered their presentations at pace, but one undeniable element that couldn't be disguised or hidden was their obvious excitement and anticipation of products nearing completion. Sure there were bugs and not all would run to plan but they were using the actual products. This was miles ahead of last year's Sydney DevCon and the product is so damn close it has a tactile presence. These three guys didn't spend significant hours of preparation time in developing a comprehensive training course complete with a 318 page manual and examples, then spend some 29 hours to travel half way round the world, just to put on a smoke and mirrors routine. How quickly is this process nearing completion? Well, hiccups that Bob Foreman unexpectedly encountered on the Monday were fixed a couple of days later.

When you have almost 50 people in a room for a week being led through all various aspects of C7 and Clarion.NET then there's plenty of opportunity for negative attitudes to be formed, if those attitudes are justified. I never at any time even heard a snicker. All comments I did receive regarding the presentations were extremely positive. Importantly people didn't just like what they saw and experienced simply because they'd already paid for it. I know this by the fact that a number of fence sitters have subsequently put in their orders for C7 and/or Clarion.NET as appropriate.

I'm absolutely thrilled with how the conference turned out. SoftVelocity got a chance to show a rather privileged group of Aussies the early results of years of hard work and long hours, and we got to see just how close that work is to fruition.

Geoff Spillane lives in Eden, a picturesque town on the far south coast of New South Wales, Australia, roughly half way between Sydney and Melbourne. He has been using Clarion since 1991 and is the Australian Distributor for SoftVelocity and Lindersoft.

Reader Comments

Add a comment

Clarion Magazine

SQL Processing and Class Interfaces

by John DelaTorre

Published 2008-06-30

Attached to this article is some source code showing how to process a generic SQL statement using Clarion object interfaces. The main approach is to divide the process into two distinct steps: (a) process an SQL statement and (b) display the results.

I have created QueryClass that process the SQL and ResultsClass to show the result. Here's the code that creates and calls those classes:

```
oQuery      &QueryClass
oResults    &ResultsClass  !Inherits from QueryClass
SQL         CString(1024)  !The SQL Query String

  Code
  SQL  = 'select client_no, name,address,zip,phone '|
    & 'from 'dbo.democlient order by name '
  oQuery &= new(QueryClass)
  oResults &= New(ResultsClass)
  oResults.Run(SQL,oQuery.SQLInterface)
  Dispose(oResults)
  Dispose(oQuery)
```

QueryClass saves the results into a memory queue and passes the memory queue to ResultsClass which displays the queue items.

A few key issues to consider:

1. The two objects are independent of each other. They do not share common data structures, and they only pass requests or responses through the interface method.
2. The application that invokes these two objects neither shares global data structures with the two objects nor invokes any code that would aid in the functioning of each object. Its sole purpose is to instantiate these two objects and destroy them afterwards.

The main point is that the call to oResults.Run doesn't pass the oQuery object, it passes the interface implemented by QueryClass. Here's the definition of that interface:

```
SQLInterface   Interface
ExecSQL        Procedure(*Cstring,ResultsQueue) Long
               End
```

This interface has one method, ExecSQL. Here's how the ResultsClass.Run method is declared so it can receive the interface:

```
ResultsClass.Run Procedure(*Cstring p:SQLStr,|
                          SQLInterface sqlIFace,|
                          ResultsQueue ResultsQ)
```

Because ResultsClass.Run expects the interface, it's not restricted to just QueryClass or classes derived from QueryClass but can work with any class that implements SQLInterface.

Virtual tables

The QueryClass.SQLInterface.ExecSQL method uses a "virtual table" called TempSQL to execute an SQL query using Prop:SQL. The ExecSQL code processes TempSQL like a regular Clarion file. The issue is that TempSQL must match its structure with the returned results that will be filled into a queue. But the query may have variable columns and the queue structure can not be predetermined at runtime.

I did try to use a class queue reference to store the results, but couldn't get this to work without the queue being empty and the app crashing. The critical solution was to pass a reference variable of type ResultsQueue. Here's the code for the ExecSQL method:

```
QueryClass.SQLInterface.ExecSQL Procedure(*Cstring p:SQLstr,ResultsQueue ResultsQ)                    !p:
SQLStr = The SQL string passed as parameter. Trying to do encapsulation
TempSQL  FILE,DRIVER('MSSQL','/LOGONSCREEN=FALSE,/SAVESTOREDPROC=FALSE'),|
  OWNER('<server>,<database>,<database>,<password>'),|
  NAME('dbo.DemoClient'),PRE(xql)
Record     RECORD,PRE()
Col1        CString(1024),NAME('Address')
Col2        CString(1024),NAME('Address')
Col3        CString(1024),NAME('Address')
Col4        CString(1024),NAME('Address')
Col5        CString(1024),NAME('Address')
Col6        CString(1024),NAME('Address')
Col7        CString(1024),NAME('Address')
          End
         End
RetVal   Long
 Code

 Free(ResultsQ)
 Share(TempSQL)
 If ErrorCode()
   Message('Share() Error: '&Error()&' : '&FileError(),|
     'SQL     Error',icon:Exclamation)
   RetVal = ErrorCode()
 End!

 If ~RetVal
   TempSQL{prop:sql} = Clip(p:SQLStr)
   If ErrorCode()
     Message(Clip(TempSQL{prop:sql})&' error: '& |
       Clip(FileError())& ': '&FileErrorCode(),|
       'SQL Error',icon:Exclamation)
     RetVal =  ErrorCode()
   End!
 End!

 Loop
   Clear(xql:record)
   Next(TempSQL)
   If ErrorCode()
     Break
   End!If

   ResultsQ.record = xql:Record
   Add(ResultsQ)
   If ErrorCode()
     Message('Add(ResultsQ) Error: '&Error(),|
       'Queue Error',icon:Exclamation)
     RetVal = ErrorCode()
     Break
   End!If
 End!Loop

 Close(TempSQL)
 Return RetVal
```

ExecSQL declares a TempSQL file definition which uses the technique of mapping its columns to a single field in an actual table (using the NAME attribute) while retrieving data from a different table. The SQL statement returns the result set in order of the SELECT statement, and since the field structure in temporary SQL table matches the ResultsQueue declaration I can simply copy the data across in bulk.

If I needed to change the implementation of the query class, say to ADO or some other approach, I would just create another class that implements SQLInterface and use that one instead of QueryClass.

The only limitation to this temporary table approach is that I cannot use a query like 'SELECT * FROM ...' because the column heading is defined by parsing the query string and QueryClass does not interrogate any catalog from the backend to determine the structure of the table.

The program also illustrates the use of SQL, files, records, queues, groups, classes, windows, etc., in a project environment without using Clarion ABC, templates, apps, and dictionaries. This is pure hand-coded Clarion; I hope it helps illustrate the real power of the Clarion language.

Download the source

Reader Comments

Posted on Monday, June 30, 2008 by Gustavo Schiaffino

May be the use of dynamic drive is a god solution to the temp table ?

Add a comment

Clarion Magazine

Single Sign-on Using ActiveDirectory

by Nardus Swanevelder

Published 2008-06-27

Recently Marty Honea wrote an article called Querying ActiveDirectory In Clarion. It came at exactly the right time for me as I have a client that asked me to implement single sign-on in one of my applications.

What do I mean by single sign-on? The client wants me to check if a user exists in the Microsoft ActiveDirectory (AD) and if the user is not disabled in AD I must log that user into my application without any other request for user IDs and/or passwords.

I have explained the security risks associated with this method to the client but the client insists that they want to use the single sign-on method. As the client is always correct I had to find a way to accomplish this. In all fairness the advantage of using single sign-on is that the administrator only has to disable a user in one place and that user won't have access to any one of the systems running on the entire network. This reduces the administrative workload as well as the risk of a user leaving the company and still having access to systems on the network.

To add to the problem I am also using CapeSoft's SecWin product which means that I already have a user ID and password for everyone who will be using my application. This means that I will have to map each AD user to each SecWin user. To do this I created a mapping table in my dictionary.

In my case it is not possible to take SecWin out of the equation as SecWin is in control of my User profiles and Security. I understand that there are different security add-on products out there, so if you're using one of them you will have to replace the SecWin code with your own product's code. Please see the ClarionShop website for a list of Security Add-On products.

After reading Marty's article and understanding more of AD, I was of the opinion that the following process should solve my problem:

1. Get Active Windows UserName.
2. Check if that UserName exists in my Mapping Table.
3. If the User exists in SecWin, check if the UserName is a valid AD User and check that the user is not disabled in AD.
4. If the User is a valid AD User and the User is not disabled, log the User into SecWin.

I started with Marty's example and quickly ran into some problems. I could not connect to my AD. After some investigation and displaying the error code returned by dbConn._Execute I found that error 2147217865 means that ADO can't find a required table. For a complete list of errors please see the article List of Errors Returned by ADOFiltr.dll.

The first step was to check if AD was accessible from my PC. Although I could log on to the domain I wanted to make sure that I didn't have any access issues. This lead me to a utility called PortQry. For a complete overview on this utility go to How to Use PortQry to Troubleshoot Active Directory Connectivity Issues. In short this utility give you details on the sequence of events that takes place while connecting to AD.

After using this to test the access to ActiveDirectory I found that I had to change my LDAP settings from

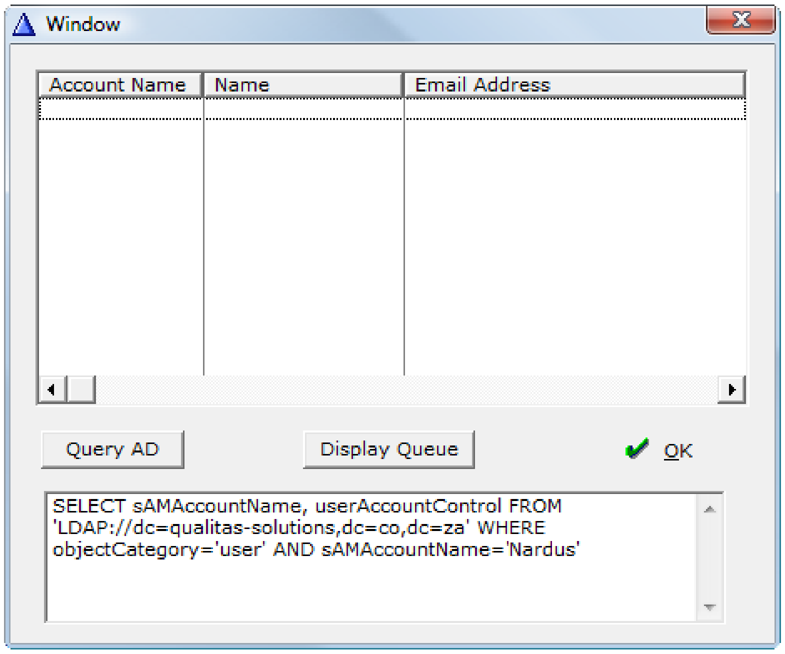
```
<<LDAP://DC=YourDomainName, DC=com>
```

to

<<LDAP://DC=YourDomainName, DC=co, DC=za>

This is due to the fact that my domain is not YourDomainName.com but YourDomainName.co.za. In hind sight I should have picked this up quickly but due to my lack of knowledge on AD I thought that DC=com was indicating where you had to search in AD.

After fixing the domain setting I could connect to AD without any problems. Now I started playing with different search strings to see what information I could get back from AD. To make this process easier I modified Marty's example application so that I could change the search string at runtime. I also added functionality to run through the result queue and displaying the different values of each entry in the queue (Figure 1).



```
SELECT sAMAccountName, userAccountControl FROM
'LDAP://dc=qualitas-solutions,dc=co,dc=za' WHERE
objectCategory='user' AND sAMAccountName='Nardus'
```

Figure 1. ADO Connect Screen

Let's look at Marty's search string in more detail:

<<LDAP://DC=YourDomain,DC=com>;

This is telling LDAP the domain to which you are connecting; I have covered this in some detail above.

```
'(& (objectCategory=person)(objectClass=user)' |
  &'(userAccountControl:1.2.840.113556.1.4.803:=2));
```

This part is specifying which object you are searching for. One other example is as follows (objectCategory=user):

```
uid,' |
  &'sAMAccountName,cn,sn,name,name,givenname,title,description,' |
  &'department,legacyExchangeDN,telephoneNumber,mail,lastLogon,' |
  &'lastlogoff,logonhours,pwdlastset,accountexpires
```

This set of parameters is all the parameters that the search will return. It is important to make sure that the parameters that will be returned also exist in the ResultQ definition, otherwise the returned parameters will not be mapped to the result queue by the Mapper.MapRsToGroup(Rs,ResultQ) method. The parameter names also have to be based on Microsoft's definition. For example you have to use sAMAccountName; if you use SAMAccount you will get an error code 2147217900 which means that the command you are sending is incorrect. Check this link for a list of all the attributes.

Lastly, subtree specifies that you want to search the entire AD sub tree.

For more information on the search string go to Microsoft OLE DB Provider for Microsoft Active Directory Service.

Here are some other search strings to give you a feeling for possible solutions.

Search for all disabled users in a co.za domain:

```
SqlStr = '<<LDAP://DC= YourDomain, DC=co, DC=za>; ' |
  & '(&(objectCategory=person) (objectClass=user) ' |
  & '(userAccountControl:1.2.840.113556.1.4.803:=2)); ' |
  & 'uid,sAMAccountName,name,givenname,cn,' |
  & 'legacyExchangeDN,telephoneNumber,mail;subtree'
```

Search all the users, active and disabled

```
SqlStr = '<LDAP://DC= YourDomain ,DC=co, DC=za>; ' |
  & '(objectCategory=user); uid,sAMAccountName,name,' |
  & 'givenname,cn,legacyExchangeDN,telephoneNumber,' |
  & 'mail,userAccountControl;subtree'
```

SQL Selects

The following searches make use of a "standard" SQL Select statement. All the information specified in Marty's search string is also specified in the Select statement, it is just specified in different places. I prefer using the Select statement but that is because I am working with MS SQL and the Select statement is not foreign to me.

Search for a specific user based on given name and surname:

```
SqlStr = 'SELECT sAMAccountName, Name, userAccountControl ' |
  & 'FROM "LDAP://dc=qualitas-solutions,dc=co,dc=za" ' |
  & 'WHERE objectCategory="user" AND ' |
  & 'givenName="Nardus" AND sn="Swanevelder"'
```

Search for a specific user based on the Windows login name:

```
SqlStr = 'SELECT sAMAccountName, userAccountControl ' |
  & 'FROM "LDAP://dc=qualitas-solutions,dc=co,dc=za" ' |
  & 'WHERE objectCategory="user" AND ' |
  & 'sAMAccountName="' & Clip(GLO:LoginName)  & '"'
```

There are many parameters that AD can return but for my purposes I am only interested in two. The first one is sAMAccountName which is the user's login name, and the second one is userAccountControl, which among other things contains information about the user's account status. See How to use the UserAccountControl flags to manipulate user account properties for more information on the UserAccountControl flags.

Now that you have a better understanding of the search strings it is time to start looking at the code to see what I have done to get my single sign-on process to work.

Inside the global map you have to add the following code:

```
INCLUDE('CWUTIL.INC'),ONCE
```

Adding this include gives you access, among other things, to Clarion's GETUSERNAME() method. You use this method to determine the active Windows user name, which you can use to check if the user is a valid AD user; if so get the SecWin login details.

I made use of Marty's code with only a few changes. One of the biggest changes was to move Marty's code to the Main procedure as this is where I am testing the login information. Please see the downloadable source zip at the end of this article for an example.

I added the following code to the Init embed point of the Main procedure. Please read through it, it is fairly self-explanatory.

```
!Get Active Windows UserName
GLO:LoginName = GETUSERNAME()
!Check LoginName against SystemUser Table
SYSUSE:sAMAccount = GLO:LoginName
If Access:SystemUsers.TryFetch(SYSUSE:SK_SAMAccount) |
  = Level:Benign
  !Found LoginName in System Table, now look for
  ! LoginName in AD
  !Get the AD LoginName (sAMAccountName) and the
  ! UserAccountControl info from AD by
  ! looking in the "qualitas-solutions.co.za" domain
  SqlStr = 'SELECT sAMAccountName, userAccountControl ' |
    & 'FROM "LDAP://dc=qualitas-solutions,dc=co,dc=za" WHERE '|
    & 'objectCategory="user" AND sAMAccountName='' |
    & Clip(SYSUSE:sAMAccount)  & ''

    !You have populated the search string, now connect to AD
    ! and do the Search
    Do Check_UserInActivedirectory

    !Check User Info Returned by AD
    Get(ResultQ,1)
    If not error() and ResultQ.sAMAccountName = SYSUSE:sAMAccount

      !Check if User is disabled
      !0010b = 2 which is setting if user was disabled
      If Band(ResultQ.userAccountControl,0010b)
        Message('User is Disabled in Active Directory')
        Return Level:Cancel
      Else
        !Valid AD User
```

```
          !Replace the following code with your own
          ! Security product s code

          !Login to SecWin

          !SecLoc:AreaName = Clip(AppNameDesc) & ' | Main'
          !SecLoc:Options     = MyOptions#+DS_CASESENSITIVITYOFF
          !SecLoc:User         = SYSUSE:UserLogonId
          !SecLoc:Password = SYSUSE:UserLogonId
          !AppNum = ds_LoginUser(SecLoc:UserLoginOptions)
          !If AppNum = 0
          !   Message('Not a valid SecWin User')
          !   Return Level:Cancel
          ! Else
             !Valid SecWin User
          !  End
        End
      Else
        Message('Not a valid Active Directory User')
        Return Level:Cancel
      End
    Else
      Message('Not a valid System User')
      Return Level:Cancel
    End
```

I need to point out to the SecWin users that I will create the users in SecWin with the password being the same as the user ID. I will not allow the system users to change their passwords as the password control is now handled by AD.

I am not using SecWin's user password authentication anymore but I am still using all of the other SecWin functionality. This means that if a user is created in AD that I will have to add that user in SecWin as well. In SecWin I create the user with the password identical to the user ID to make login into SecWin a seamless process. It is important to note that the AD user ID and the SecWin user ID don't have to be the same as the mapping table maps the two to each other.

For example I might have a user ID Nardus.Swanevelder in AD and a user ID NardusS in SecWin and my mapping table will map them together. When the user tries to access my system his Windows user name will be Nardus.Swanevelder which will be authenticated against the AD database; after that I will do a fetch against the mapping table and then log user NardusS with password NardusS into SecWin.

The same principles would apply to any other third party security product that you might be using.

In the attached source I haven't included the SecWin code so that those users without SecWin can adapt the code to their preferred product.

Summary

I have implemented single sign-on functionality in my application by making use of an ADO query to determine the validity and account status of a user in the Microsoft's Active Directory. I also use this information to log the user into my third party security product of choice, thereby creating the illusion of a single sign-on.

Download the source

Nardus Swanevelder was born and raised in South Africa. He was a networking engineer for seven years before he moved over to the commercial side of the business. Nardus has developed a Sale Cycle Management system for the Information and Communication Technology industry. He has been programming in Clarion since 1989, and holds B.Com and MBA degrees. In his spare time Nardus lectures Financial Management to B. Com Hons students at North-West University.

Reader Comments

Posted on Friday, June 27, 2008 by arne skov

Very nice work and very nice references to further information (ADO errorcodes and Microsoft definitions). Thanks.
Never the less I cannot figure out how to port this example to accessing an LDAP server on a Linux box.

Posted on Friday, June 27, 2008 by Nardus Swanevelder

This article is about interfacing to Microsoft Window's ActiveDirectory using a LDAP query via Clarion's ADO.

I have not tested LDAP running on a Linux box but based on some information on the WEB it should be a case of changing the DC settings to point to the domain name of your Linux box and making sure that the attributes you are requesting are supported by LDAP running on the Linux box.

You will have to check the LDAP server on the Linux box to see which properties are supported. For e.g. it might support cn, sn and mail but not sure if it supports sAMAccountName. It looks like you will have to use uid and not sAMAccountName.

I have no idea if LDAP on Linux has a equivalent for Microsoft's userAccountControl. You might have to add this to the Active Directory Schema on your Linux box.

Some useful links:
http://wiki.freaks-unidos.net/linux%20ldap%20howto
http://www.yolinux.com/TUTORIALS/LinuxTutorialLDAP.html
http://tldp.org/HOWTO/html_single/LDAP-HOWTO/

LDAP Software
http://www.openldap.org/
https://opends.dev.java.net/

Don't know if this answer your question but maybe you can write an article for ClarionMag on how to do LDAP to a Linux box. :-)

Add a comment

Clarion Magazine

Running Clarion On Vista 64

by Mark Riffey

Published 2008-06-26

I have a 64 bit Vista machine on which I need to run Clarion 6.3. But 6.3 uses the old 16 bit IDE, and 16 bit applications aren't supported by 64 bit Windows. The answer is something called virtualization software which lets you install multiple versions of Windows on one machine and run them at the same time.

First I tried Microsoft Virtual PC. The price was right (free in the U.S.). It was very easy to get a virtual machine set up and running and install XP Pro on it.

I installed Clarion6.3.9050 from the CD, opened and entered my CD key and closed Clarion. Next, I copied my current development environment to the same folder. Network performance was surprisingly good for Virtual PC. Clarion seemed to work briefly, but then I kept getting GPFs when loading an app.

I had set aside 3 gig of ram for this XP session (close to the maximum for a 32 bit machine), so I wasn't memory constrained. I set Virtual PC aside for a day to get some real work done and then decided that I needed to try VMWare.

VMWare Server is free. Unfortunately, the device drivers are not signed so you can't install it on Vista64, since it requires that all devices drivers are signed. There are workaround for this, but I think they are foolish hacks so I won't even discuss them.

So I bit the bullet and decided to download the 330 megabyte trial version of VMWare Workstation, which costs $189 when downloaded. I requested activation, installed VMWare Workstation with the supplied key, and rebooted Vista64. After the reboot I started VMWare Workstation and was prompted to create a virtual machine. I did so, giving it 20GB of disk space and 2 gigs of RAM (the same as my Vista32 laptop).

I popped in an MSDN DVD that contained XP Pro Service Pack 1 and I told VMWare Workstation to start the VM. VMWare runs each virtual machine in its own window. First the BIOS messages appeared, and then up popped a menu for me to choose which OS to install (this was from the DVD, not VMWare). The XP install was standard stuff, nothing odd or special for me to do, other than noting in the VMWare Guest install docs that I shouldn't activate XP until after installing VMWare Tools on the guest OS.

Differences

Virtual PC will not let you access a network drive as a shared folder. VMWare not only will allow the shared network folder, it will also let you map a drive letter to it. The downside of this is that shared folder performance is really bad inside of a virtual machine, at least it was when I first tried it. There are a few tips for dealing with this, including disabling VMWare's virtual network adapters. That seemed like a great way to cause a problem that would fill a day or so, so I blew that off.

After googling around a little bit I restarted the XP VM and network performance improved. Later, I saw that network access was typically better with a few sluggish moments when reading from my NAS drive. Once again I installed Clarion6.3.9050 from the CD, opened and entered my CD key and closed Clarion. Next, I copied my current development environment (6.3.9056) a folder on the virtual machine.

This time around everything worked fine and performance was good.

After a week, I've had none of the problems that occurred with Virtual PC and I get the best of both worlds (such as being able to run 64 bit Lightroom, among other applications).

Finally, another cool bonus. Because the XP Pro VM running Clarion is indeed another machine I no longer have to beware of the lost focus embeditor issue when I switch to another application - unless of course I use the VM session for the other application (and I don't).

Mark Riffey has been in the software business in one form or another since the early 1980s. His background includes software development for two internationally known enterprise software vendors, the world's premier information systems services company and a Fortune 100 manufacturer. After leaving the ranks of the gainfully employed, he bought a nearly-dead vertical-market software company and grew it to the award-winning market leader in its niche. After selling that company in 2005, he now runs Rescue Marketing, Inc., a marketing, business and technology advisory firm. His business philosophy is simple: Be fair to your customers and yourself; surround yourself with brilliant people who can take a joke, work hard, be a good listener and have a little fun. Mark and his wife Jacki have two boys, Alex and Jonathan. Mark's other interests include Boy/Cub Scouting, backpacking/hiking and almost anything else outdoors, classic blues guitar, golf and photography.

Reader Comments

Posted on Saturday, June 28, 2008 by Michael Holland

I've been using Vmware to run Windows virtual machines on Linux hosts for the past three years. By minimizing the amount of stuff in any given Windows virtual machine you get a more stable and longer lasting environment. It's a nice way to be able to develop using multiple versions of Clarion on one machine, or look at one Clarion AP in one virtual machine while developing on another.

Add a comment

Clarion Magazine

Aussie DevCon Day 4 Training

by Geoff Robinson

Published 2008-06-25

The fourth day of training was the final one before the Aussie DevCon conference proper started.

The first presentation of the day was by Robert Zaunere. (I realize a lot of people call RZ "Bob" but I will use the more formal "Robert" throughout to save any confusion with Bob Foreman.) This was Robert's first presentation apart from the welcome on the first day. The topic covered was Going Mobile - we looked at the challenges in designing an efficient mobile user interface using the .NET Compact Framework and Clarion.NET.

The .NET Compact Framework is a subset of the .NET Framework designed for "resource-constrained" devices like PDAs and Smart mobile phones. You need the Enterprise version of Clarion.NET in order to use the Compact Framework development tools.

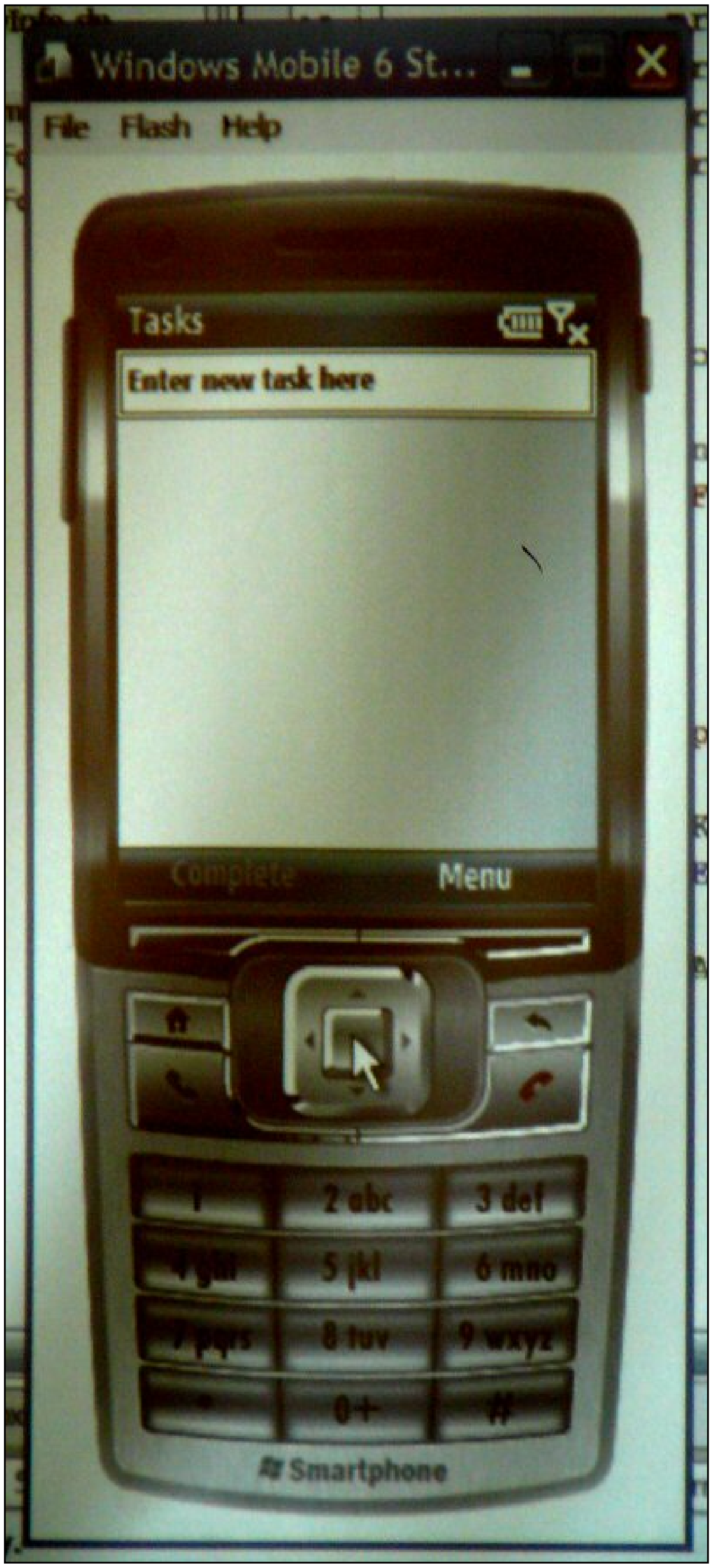
.NET Compact Framework 2.0 Windows Forms Controls

Robert said that we already knew 90% of what we needed to know to develop for mobile devices and he would show us the remaining 10%.

As with other .NET apps there are three possible ways to create Clarion# mobile application: wizards, code templates (quickstarts) and apps. The template set that will come with the Clarion.NET application generator is still being developed, and at the moment the DNET template wizard (that takes file definitions from your dictionary and uses C6 to generate an application) only creates desktop apps (although you might be able to adapt these, with the caveat noted below). Another possibility is to use the QuickStarts that are now included with the new IDE; these are available for both the Windows CE and Pocket PC/SmartPhone platforms. You need to read the documentation to make sure you have everything installed as there are several additional Compact Framework items that you need.

You need to be careful of with graphic images as different devices have different screen resolutions. In general you need to look closely at the mobile device you are targeting.

Robert used the Mobile 6 Standard emulator and said the emulators were very good and gave a realistic impression of how your apps would run on the target devices. Microsoft has renamed the SDKs for Mobile 6: The Standard and Classic SDKs are for smart phones and the Professional SDK is for PDAs with or without phones.

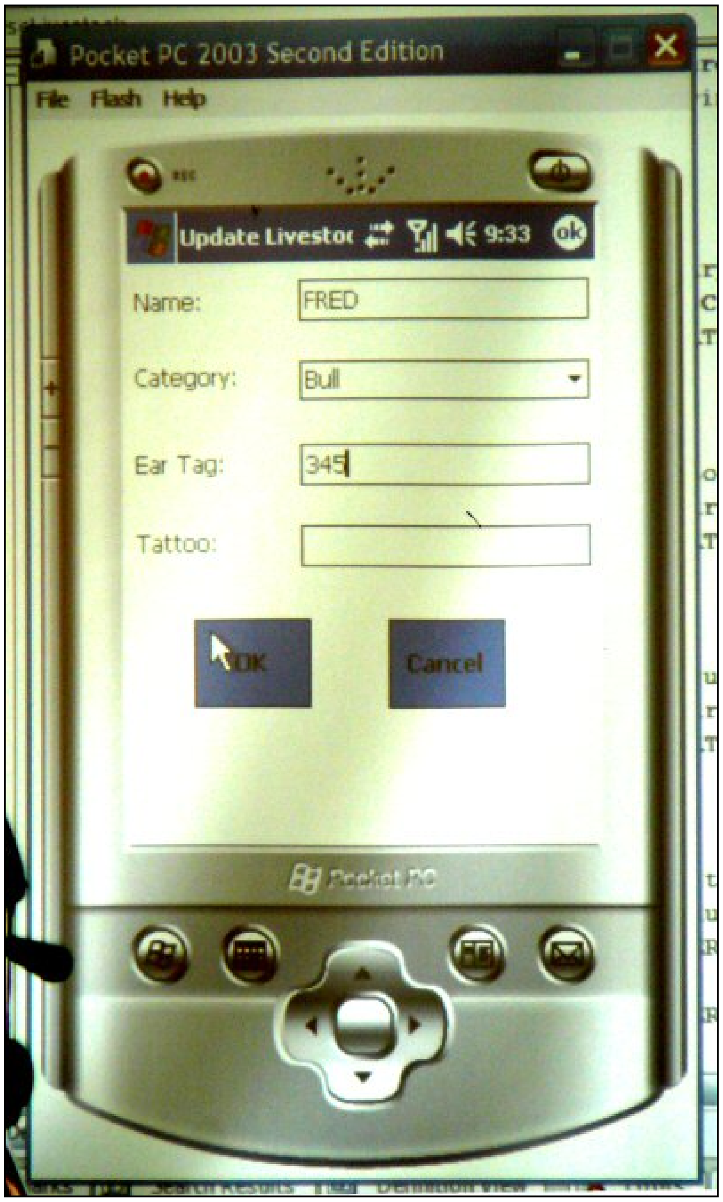


You can do rich client or thin client apps, but thin client apps which use web access via HTML controls require connection to the internet so are perhaps not used as much. With rich client apps the data is validated and stored locally, whereas with thin client this is all done on the server so there are frequent round trips.

Tabs can appear in different positions on different devices and are not supported on smart phones.

Robert showed us an example using SQLite which he said was great - fast, reliable, portable etc. SQLite also had at least a dozen free tools available on the internet. You cannot use Clarion's file drivers yet on compact/mobile devices and so should use something like SQLite. Alternatively you could use hierarchical XML files.

Robert showed a livestock program he had developed for Geoff Bomford. It was an amusing tale where Geoff had asked for a "live stocktake" program. In Australia a stocktake is an audit of what goods you have on the shelf of a store but Robert had thought Geoff was referring to livestock as in cattle and had developed a program accordingly. I guess it goes to show you should always check that your understanding of the client's requirements matches what they have in mind! Anyway this application allowed tracking of cattle and medication given to them (date/dosage etc.). The application was left in memory as mobile apps are slow to load.



Name: FRED
Category: Bull
Ear Tag: 345
Tattoo:

Robert mentioned that pocket PC remote file viewer was good for transferring files either way - to or from the device.

For mobile devices the user interface is critical - keep it simple, add in resolution awareness if it may be deployed on different devices. Remember that Pocket PC (and Pocket PC phone edition) have touch screens with a stylus and standard buttons, whereas Smartphones have no stylus. Generally in mobile applications you have limited screen real estate and memory. Some devices will have inbuilt QWERTY keyboards that are displayed on the screen while others may have voice commands. There is a lot of variability.

You should not port a desktop UI to a portable device - the user interface needs to be completely redesigned but you should be able to reuse the backing business logic. You need to avoid overcrowding controls and try to limit the number of clicks the user is required to make.

In terms of keeping the interface simple Robert suggested placing frequently used fields at the top of the form, showing only the most important controls, and grouping controls for related tasks. Use easily distinguished controls and icons and provide predictable and consistent control functions and provide feedback to the user - visual, sounds, clicks, status etc.

Robert talked about tab controls and panels - use these to logically group items and optimize real estate. All controls on all tabs and panels are loaded when the form is initialized.

If a process is going to take more than a second, activate WaitCursor to give feedback to the user:

```
Cursor.Current = Cursors.WaitCursor
! do some long process
Cursor.Current = Cursors.Default
```

Robert also talked about SIP (Soft input Panel), which is the optional popup on-screen keyboard used to simplify user input. Here are Robert's tips for SIP use:

- Make your application adjust control positioning if the user selects SIP
- Put controls where SIP will not hide them
- Test your app to make sure it works properly with SIP
- SIP will automatically hide itself if the user starts typing on an inbuilt or connected keyboard
- If you include a custom SIP, give users the choice to select it.

Geoff Robinson lives near the beach in Melbourne, Australia, and is an active member of the local Clarion User Group. His company, Vitesse Information Systems, specializes in software for local government. Geoff was impressed by Clarion back in the DOS days and grabbed the early betas of Clarion for Windows when they first became available; he has been using Clarion as his primary development environment ever since. When not in front of a computer Geoff enjoys listening to music, singing bass in a local choir, and spending time with his three young children.

Reader Comments

Add a comment

Clarion Magazine

Boost Compile Times By Putting The ABC Classes In Their Own DLL

by Benjamin Dell

Published 2008-06-18

This article is intended for developers using Clarion 6 and is a result of my frustration at having to spend hours per day waiting for large compilations to take place where much of the code being compiled is ABC source or other code that hasn't changed since Noah left the Ark.

Whenever you make global changes to an application Clarion recompiles all the source by default. That's because code has to be compiled differently depending on whether you're creating a LIB, a DLL or an EXE, and the IDE doesn't keep track of which way you last compiled the code.

The way to reduce this unnecessary code compilation is to isolate the ABC and other library code into DLLs and then reference that compiled code instead of the source code. In this article I'll show you how it's done.

Clarion supports the ability to create the following types of application configurations:

- Executable - Code, usually in the form of a disk file that can be run by the operating system in use to perform a particular set of functions
- Library - A library is a module that contains functions and data that is linked into a dynamic link library or an executable for use by that dynamic link library or executable.
- Dynamic Link Library - A dynamic link library (DLL) is a module that contain functions and data that can be used by another modules

No matter if I create a single EXE or multi-DLL solution, I always tend to include the ABC class definitions and my dictionary data definitions in the same EXE or DLL. Even legacy code isn't safe, since the legacy templates include much new functionality that relies on ABC.

The result of this is that I have to recompile the ABC class code every time I recompile my applications. This has advantages and disadvantages:

> PRO: I am sure that any changes to the ABC or other library classes are reflected in my applications.
>
> CON: I waste time compiling class definitions that have not changed since the last Clarion or third party product used in my software has been updated.

Typically when I work with multi-DLL applications I have one DLL to do the class and dictionary definitions, more DLLs that hold procedures and a single EXE that call these procedures. Figure 1 illustrates the scenario where all the class and data definitions are exported from the same DLL. Exported simply means that the definitions are contained in the DLL but their definitions are exported for use by other DLLs/EXEs.

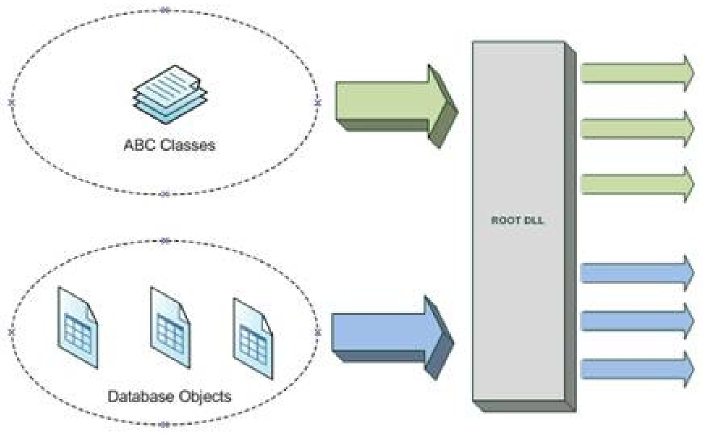


Figure 1. Exporting ABC classes and database objects from a root DLL

What I typically want to achieve is the separation of the ABC classes and the database objects, allowing me to code and compile the database objects under my control without the frustration of having to re-compile the same ABC classes over and over again with no added benefit.

The image in Figure 2 depicts my ideal situation.

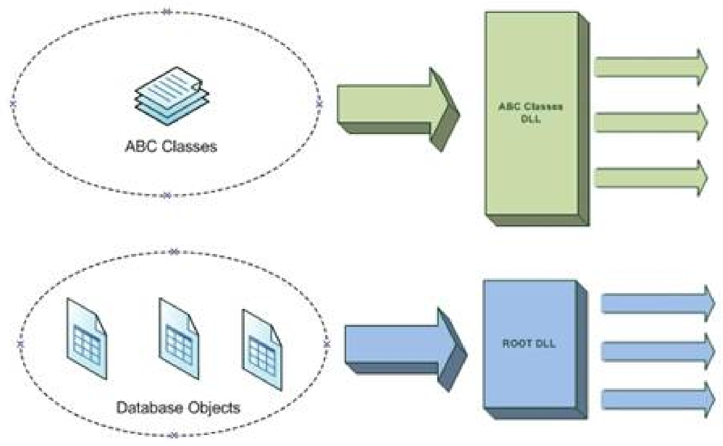
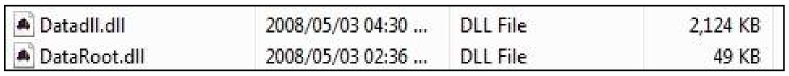


Figure 2. Separating the ABC classes

Clarion 6 (and earlier) gives me the ability, with some tinkering, to separate the creation of the ABC classes and the database object into two separate DLLs. This drastically reduces the size and compile time of my DLLs.

An example

I will use the people.app example application that comes with Clarion 6 to show how to radically reduce the size of a data DLL, as shown in Figure 3.



Datadll.dll	2008/05/03 04:30 ...	DLL File	2,124 KB
DataRoot.dll	2008/05/03 02:36 ...	DLL File	49 KB

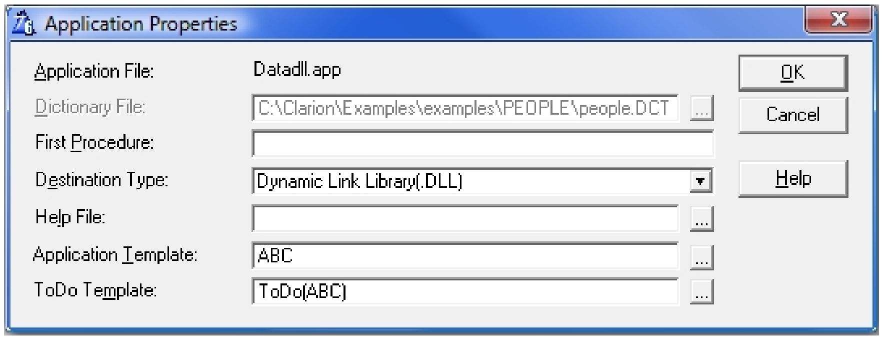
Figure 3. Reducing the data DLL's size

Figure 3 shows a datadll.DLL that holds the ABC class definitions as well as the dictionary objects. The dataroot.DLL file is the same file but with the ABC classes removed. This dramatically speeds up the application recompilation time.

What goes wrong

Why don't I have this behavior by default? The way that the ABC templates are organised and the prompts available to the programmers tend to force developers into lumping the ABC classes and the dictionary objects in the same DLL.

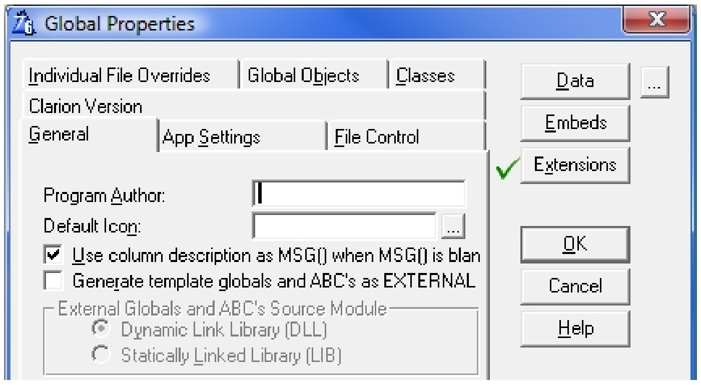
To convert people.app into a multi-DLL solution I would normally create a new data DLL app and define it as depicted in Figure 4.



Application File:	Datadll.app
Dictionary File:	C:\Clarion\Examples\examples\PEOPLE\people.DCT
First Procedure:	
Destination Type:	Dynamic Link Library(.DLL)
Help File:	
Application Template:	ABC
ToDo Template:	ToDo(ABC)

Figure 4. Creating a data DLL application

I specify the details for my data DLL on the global properties as depicted in Figure 5.



Program Author:

Default Icon:

[✓] Use column description as MSG() when MSG() is blan

[ ] Generate template globals and ABC's as EXTERNAL

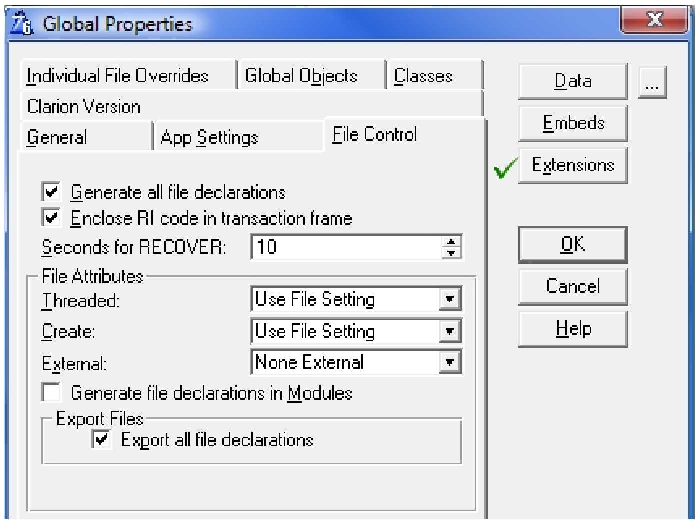
External Globals and ABC's Source Module

(●) Dynamic Link Library (DLL)

( ) Statically Linked Library (LIB)

Figure 5. The default global properties

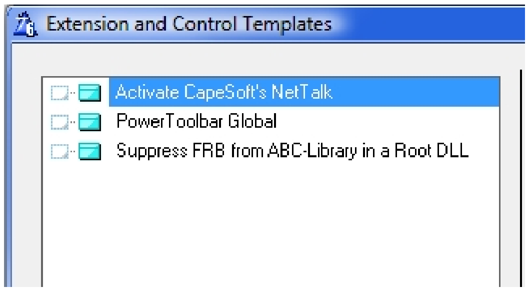
Notice that the check box under External is not ticked (Figure 6), meaning globals and ABC objects are defined inside this app file.



[✓] Generate all file declarations
[✓] Enclose RI code in transaction frame
Seconds for RECOVER: 10
File Attributes
Threaded: Use File Setting
Create: Use File Setting
External: None External
[ ] Generate file declarations in Modules
Export Files
[✓] Export all file declarations

Figure 6. Exporting file declarations

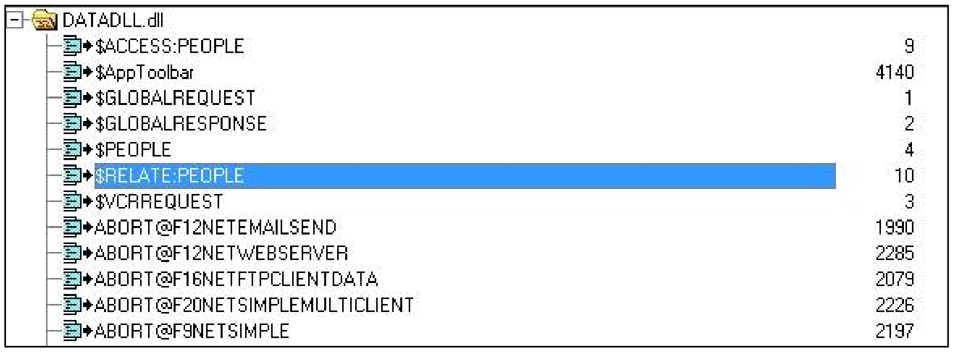
I also instruct this app file to hold the data definitions for the dictionary I specified when I created the app file. If I have third party global templates and classes I need to include, I would do it as shown in Figure 7.



Activate CapeSoft's NetTalk
PowerToolbar Global
Suppress FRB from ABC-Library in a Root DLL

Figure 7. Adding third party libraries to the data DLL

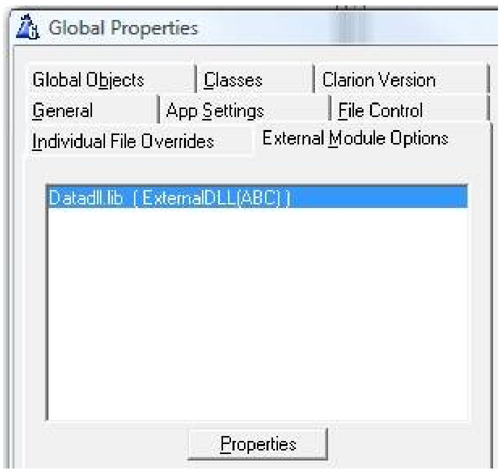
All of this will then compile and result in a Datadll.DLL file and Datadll.lib file on my hard disk. Figure 8 shows the exports from the DataDLL.



```
DATADLL.dll
  $ACCESS:PEOPLE                    9
  $AppToolbar                    4140
  $GLOBALREQUEST                    1
  $GLOBALRESPONSE                   2
  $PEOPLE                           4
  $RELATE:PEOPLE                   10
  $VCRREQUEST                       3
  ABORT@F12NETEMAILSEND          1990
  ABORT@F12NETWEBSERVER          2285
  ABORT@F16NETFTPCLIENTDATA      2079
  ABORT@F20NETSIMPLEMULTICLIENT  2226
  ABORT@F9NETSIMPLE              2197
```

Figure 8. Exports from the data DLL

When I want to use these exported objects in my People.app file I include the datadll.lib file in my external modules area and it will show up on my global settings (Figure 9) and in my module tree of my people app (Figure 10).

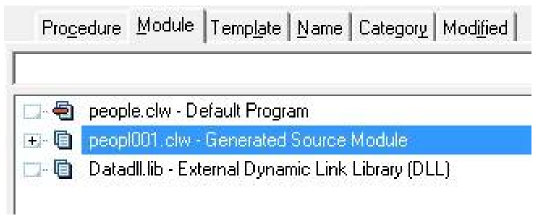

Global Properties

Global Objects | Classes | Clarion Version
General | App Settings | File Control
Individual File Overrides | External Module Options

Datadll.lib (ExternalDLL(ABC))

Properties

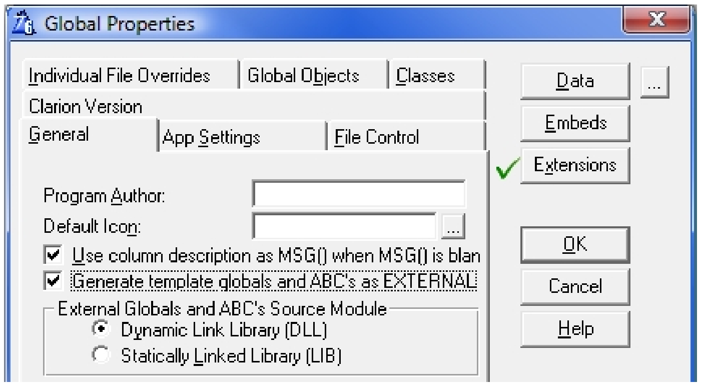
Figure 9. The data DLL in Global Properties


Procedure | Module | Template | Name | Category | Modified

people.clw - Default Program
peopl001.clw - Generated Source Module
Datadll.lib - External Dynamic Link Library (DLL)

Figure 10. The data DLL in the module tree

For my people application to compile correctly I will now change the settings on my global templates in my people application so global variables and ABC classes are declared with the EXTERNAL attribute (Figure 11). Without this change I'll get duplicate symbol errors as the compiler will find both the locally declared variables and those in the data DLL.


Global Properties

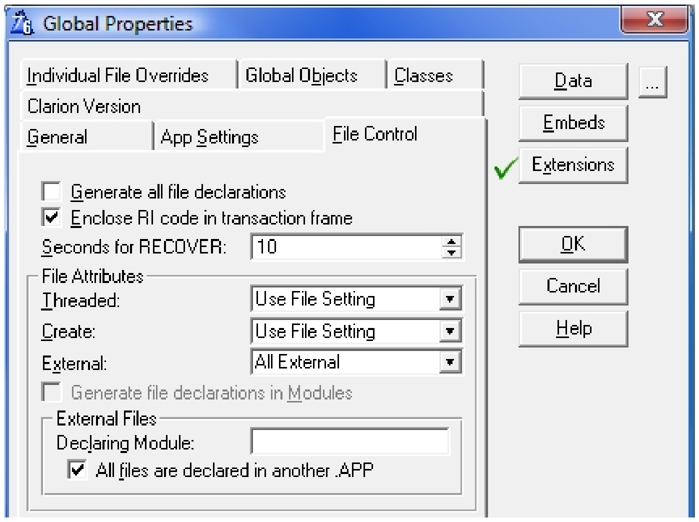
Individual File Overrides | Global Objects | Classes
Clarion Version
General | App Settings | File Control

Program Author:
Default Icon:
[✓] Use column description as MSG() when MSG() is blan
[✓] Generate template globals and ABC's as EXTERNAL
External Globals and ABC's Source Module
(●) Dynamic Link Library (DLL)
( ) Statically Linked Library (LIB)

Data | Embeds | Extensions | OK | Cancel | Help

Figure 11. Declaring global data as EXTERNAL.

I also need to tell the app that the files are all defined externally (Figure 12).



☐ Generate all file declarations
☑ Enclose RI code in transaction frame
Seconds for RECOVER: 10
File Attributes
Threaded: Use File Setting
Create: Use File Setting
External: All External
☐ Generate file declarations in Modules
External Files
Declaring Module:
☑ All files are declared in another .APP

Figure 12. Declaring files as external

Compiling the people.app file now will allow me to make use of the datadll.DLL file and I have my multi-DLL application.

Whenever I make a change in the dictionary, however, I have to re-compile the datadll.app. It takes anything from 7 to 35 seconds to recompile and link the datadll.DLL, depending on the compiler's decision to re-compile for all or some of the class definitions as well.

If I remove the class definitions out of the DLL (effectively creating the dataroot.DLL with only the dictionary objects and no ABC classes), it takes up to two seconds to re-compile the DLL. Remember this is a dictionary definition with only one (1) table in it.

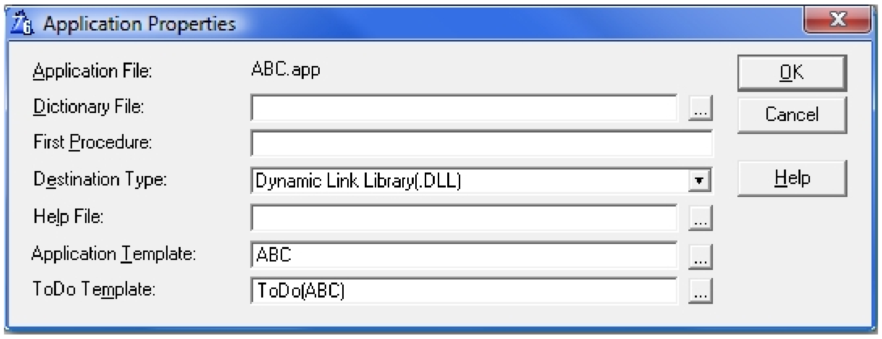
- Data DLL with Dictionary and ABC classes: 7-35 seconds to compile
- Data DLL with Dictionary: < 2 seconds to compile

Now I get to the fun part, separating the ABC classes.

Separating the ABC classes

First, I still have to create a DLL to hold the ABC class definitions but I don't have to include any dictionary information with it. I will call it ABC.DLL and as such I create ABC.app and refrain from selecting a dictionary (Figure 13). (If you get a message that a dictionary is required go to Setup | Application Options and uncheck Require a dictionary.)

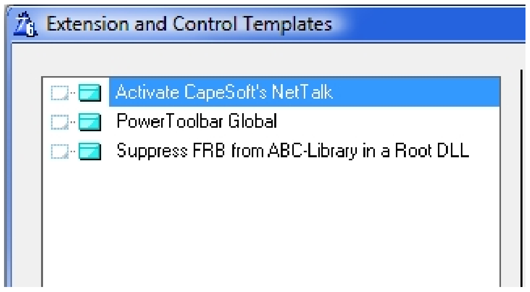
A side effect of creating a dictionary-less application is that this ABC.DLL, once compiled, can be re-used in other applications, making it an ideal global component.



Application File: ABC.app
Dictionary File:
First Procedure:
Destination Type: Dynamic Link Library(.DLL)
Help File:
Application Template: ABC
ToDo Template: ToDo(ABC)

Figure 13. Creating an ABC application with no dictionary

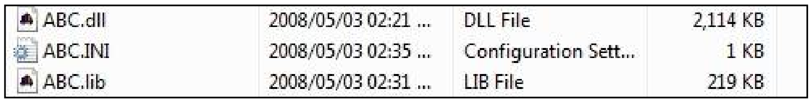
I will allow the Clarion6 IDE to generate the app for me and then I just compile it. I don't need to add any code, but if I have any third party classes and global templates, I need to include them in this app global settings area (Figure 14).


Extension and Control Templates

- Activate CapeSoft's NetTalk
- PowerToolbar Global
- Suppress FRB from ABC-Library in a Root DLL

Figure 14. Adding third party utilities to the ABC DLL.

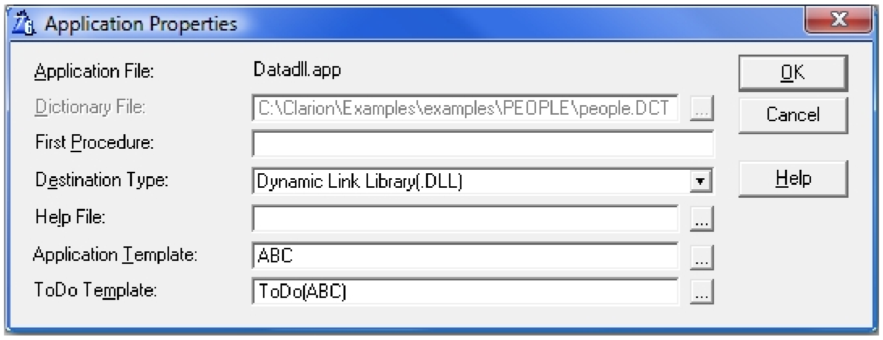
I exclude all debug information from the project for the smallest possible DLL. This will result in me having a ABC.DLL and ABC.lib file on my hard disk as shown in Figure 15.



ABC.dll	2008/05/03 02:21 ...	DLL File	2,114 KB
ABC.INI	2008/05/03 02:35 ...	Configuration Sett...	1 KB
ABC.lib	2008/05/03 02:31 ...	LIB File	219 KB

Figure 15. The ABC DLL and LIB.

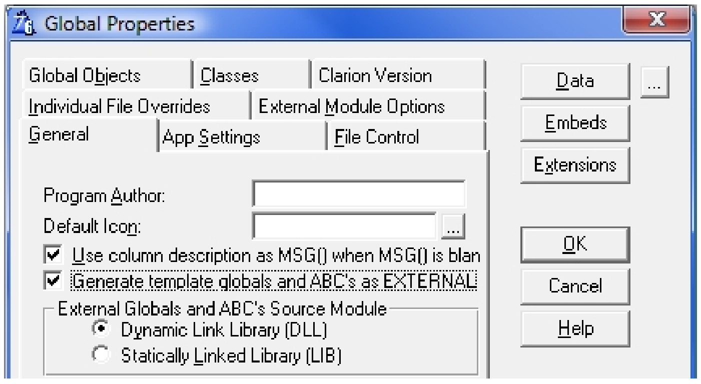
I also have to create my dictionary DLL (Figure 16), and this one I will name Dataroot.app. This will create and export all the tables I'll need. It will also be much smaller in compiled size and will compile much faster than the traditional data DLL as described in the first part of this document.


Application Properties

Application File:	Datadll.app
Dictionary File:	C:\Clarion\Examples\examples\PEOPLE\people.DCT
First Procedure:	
Destination Type:	Dynamic Link Library(.DLL)
Help File:	
Application Template:	ABC
ToDo Template:	ToDo(ABC)

Figure 16. The data DLL

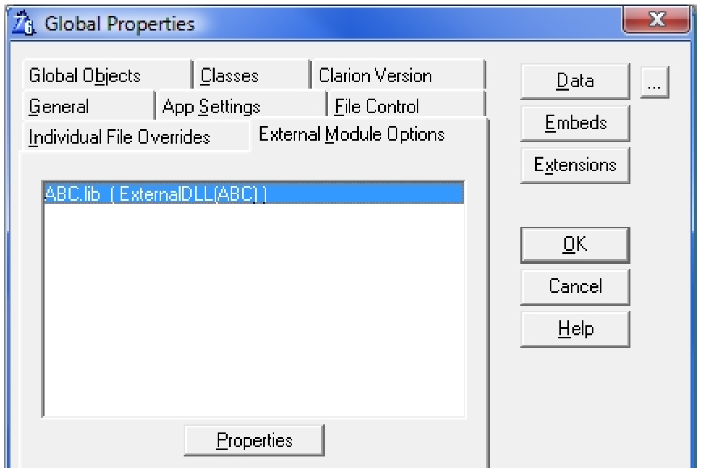
In the global settings of this application, I again have to set the Generate template globals and ABCs as EXTERNAL checkbox (Figure 17).


Global Properties

Program Author:
Default Icon:
☑ Use column description as MSG() when MSG() is blan
☑ Generate template globals and ABC's as EXTERNAL
External Globals and ABC's Source Module
● Dynamic Link Library (DLL)
○ Statically Linked Library (LIB)

Figure 17. specifying the globals and ABC declarations as EXTERNAL

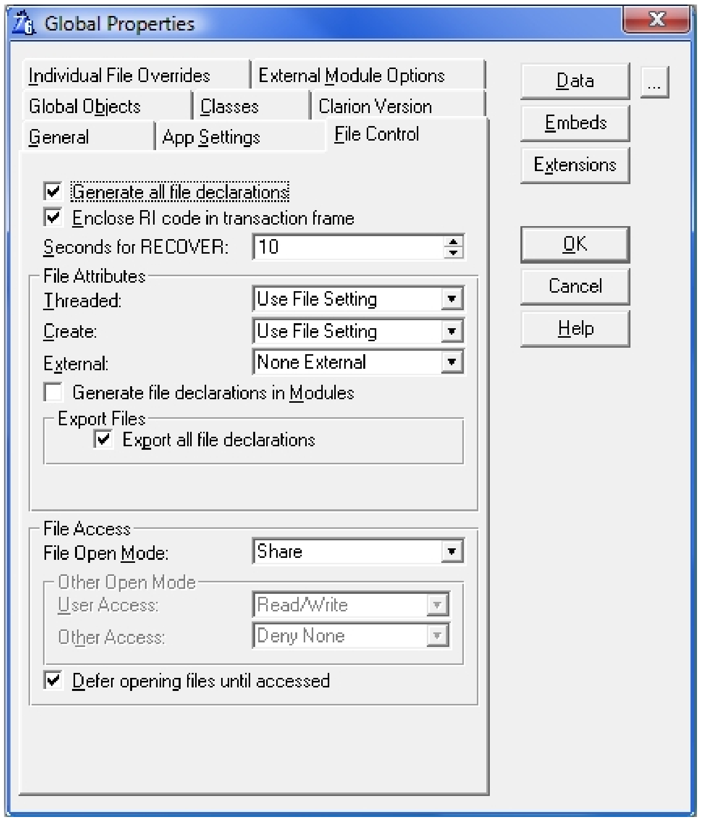
I select the ABC.DLL as an external module for my application (Figure 18).



ABC.lib (ExternalDLL(ABC))

Figure 18. Adding the ABC DLLs LIB

On the File Control tab I export all file declarations (Figure 18).



- [✓] Generate all file declarations
- [✓] Enclose RI code in transaction frame

Seconds for RECOVER: 10

File Attributes

Threaded:	Use File Setting
Create:	Use File Setting
External:	None External

- [ ] Generate file declarations in Modules

Export Files

- [✓] Export all file declarations

File Access

File Open Mode: Share

Other Open Mode

User Access:	Read/Write
Other Access:	Deny None

- [✓] Defer opening files until accessed

Figure 19. Exporting file declarations.

Normally I would compile this and expect everything to work, but due to the way that the template code is written I will get warnings of unresolved externals for all my dictionary objects.

This means that the EXTERNAL tick I did seems to bind the ABC classes and dictionary objects together...

The solution is to *untick* the EXTERNAL checkbox for globals and ABCs (Figure 20).

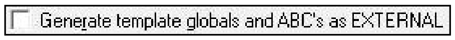
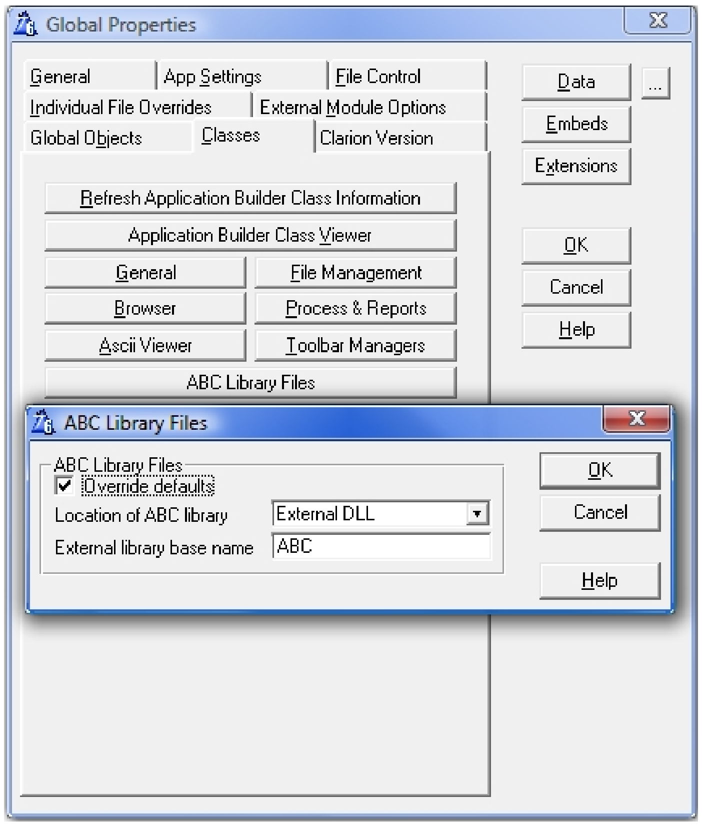


Figure 20. Unchecking template globals and ABCs as EXTERNAL

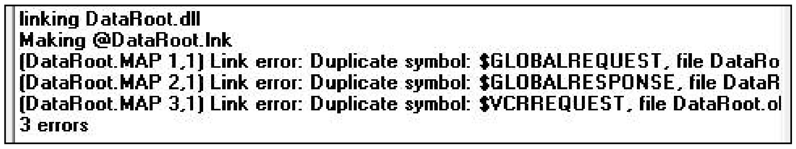
Then on the Classes tab in the global properties I specify where my ABC classes definitions are (Figure 21).



ABC Library Files
☑ Override defaults
Location of ABC library: External DLL
External library base name: ABC

Figure 21. Specifying an external library for the ABC library

Compiling this gives me three more errors on duplicate symbols (Figure 22). Although the templates let me specify an external location for the ABC library, there's no template option to do this for these three global variables.



```
linking DataRoot.dll
Making @DataRoot.lnk
(DataRoot.MAP 1,1) Link error: Duplicate symbol: $GLOBALREQUEST, file DataRo
(DataRoot.MAP 2,1) Link error: Duplicate symbol: $GLOBALRESPONSE, file DataR
(DataRoot.MAP 3,1) Link error: Duplicate symbol: $VCRREQUEST, file DataRoot.ol
3 errors
```

Figure 22. Duplicate symbol errors

And this is where the tweak come in that is non-standard, but so crucial for this method to work.

Using Libmaker (an example program distributed with Clarion 6), or Clarion Libmaker 32 from Tor Stokkan, I remove the global data definitions GLOBALREQUEST, GLOBALRESPONSE and VCRREQUEST from the ABC.lib file (Figure 23).

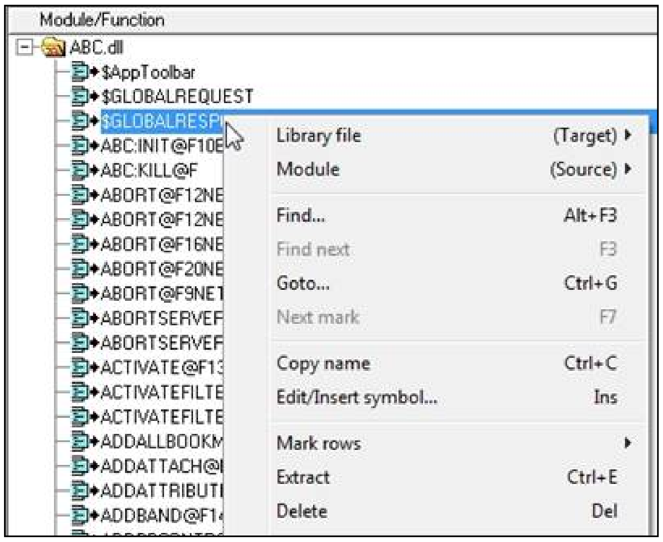


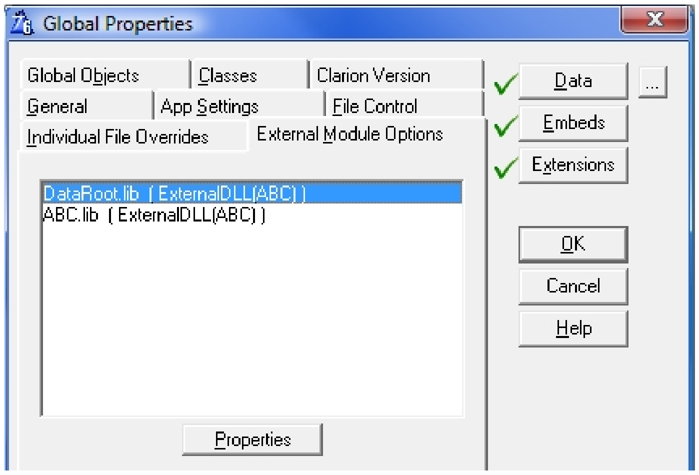
Figure 23. Removing variables from the ABC DLL.

Once I have removed the three definitions and saved ABC.lib only the versions in my dataroot.DLL will be visible.

> **NOTE:** If you are going to re-use the ABC.DLL in oher projects, you need to save the ABC.DLL file in your Clarion \bin directory and the ABC.LIB file in your Clarion\lib directory. You will also have to recompile this DLL each time Clarion is updated or a third party template or class is updated.

I can now successfully compile Dataroot.DLL.

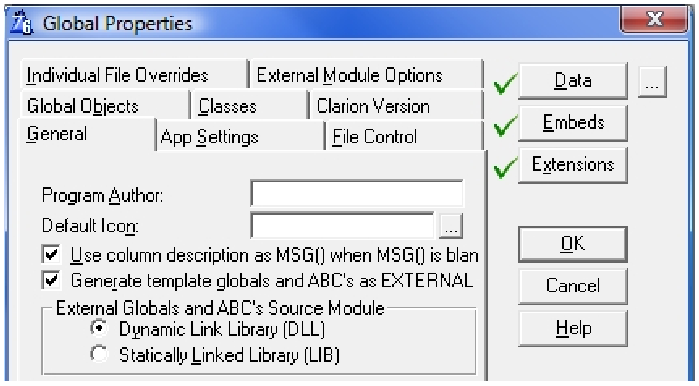
When making use of these two files (ABC.DLL and Dataroot.DLL) in other DLL and EXE files, I import them as external modules like I would do normally (Figure 24) and everything works perfectly.



DataRoot.lib (ExternalDLL(ABC))
ABC.lib (ExternalDLL(ABC))

Figure 24. Importing external DLLs

Opening the people.app file, I set the global properties to tick the EXTERNAL checkbox (Figure 25).



Program Author:

Default Icon:

☑ Use column description as MSG() when MSG() is blan

☑ Generate template globals and ABC's as EXTERNAL

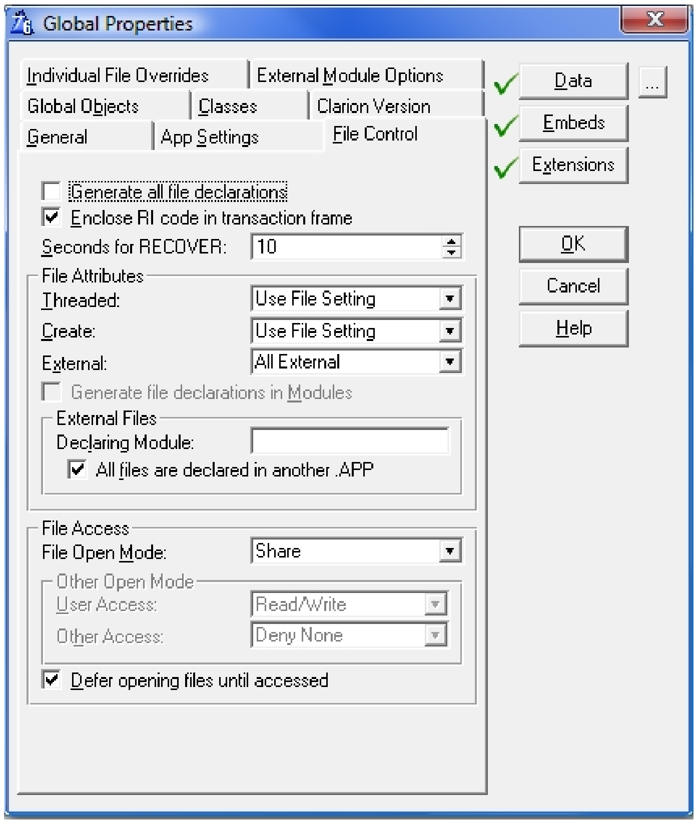
External Globals and ABC's Source Module

● Dynamic Link Library (DLL)

○ Statically Linked Library (LIB)

Figure 25. Setting template globals and ABCs as EXTERNAL

I also go to the File Control tab and set External files as declared in another .APP (Figure 26).



☐ Generate all file declarations

☑ Enclose RI code in transaction frame

Seconds for RECOVER: 10

File Attributes

Threaded: Use File Setting

Create: Use File Setting

External: All External

☐ Generate file declarations in Modules

External Files

Declaring Module:

☑ All files are declared in another .APP

File Access

File Open Mode: Share

Other Open Mode

User Access: Read/Write

Other Access: Deny None

☑ Defer opening files until accessed

Figure 26. Declaring files in another .APP

This allows me to re-compile any of the dlls and exes associated with my project, without having to re-compile the ABC classes each and every time.

Summary

Few Clarion developers make any changes to the ABC library source or to third party products, so it makes no sense to keep compiling this code over and over. By using the technique I've described in this article you can isolate the ABC library (and other libraries) from your table declarations and other globals and greatly speed up compile times.

Benjamin Dell, also known as Riebens, is a Certified Clarion Software Developer and has been using Clarion since the DOS days. He is the owner of Riebens Systems, a software contracting company in South Africa which specializez in the financial and logistic market.

Reader Comments

Posted on Wednesday, June 18, 2008 by Vince Sorensen

Considering that one original goal of the ABC classes was to avoid wasting time compiling the same code over, and over, it is amazing that an easy and obvious way to separate the classes from the data was not implemented at the start. I've wished to do that many a time. Great article! Thanks!

Posted on Wednesday, June 18, 2008 by Carl Barnes

A great idea! I have run into the issue with the way the template generated code and GlobalRexxx export conflicts when trying to create a function library DLL.

Instead of editing the LIB the issue can also be resolved by compiling once, then editing the xxx.EXP file (a text file) and removing the first 3 exports lines:

```
NAME 'XXX' GUI
EXPORTS
  $GLOBALREQUEST   @?
  $GLOBALRESPONSE  @?
  $VCRREQUEST      @?
;Start of Exported File information
```

then compiling again. The second compile should not regenerate the EXP and will use the existing one and omit the problem exports from the LIB and DLL.

A better way to perform that EXP change would be to write a global extension template that after the exports file is built (i.e. #AT %AfterGeneratedApplication) reads the EXP file back in and writes it without the 3 problem lines. That eliminates the manual edit. With the size of that EXP file it would slow down the compile a bit, but be much faster then manual editing.

Obviously the best option is to just tweak the standard templates to handle this all properly, and then continue to tweak them every release. Not an issue for 5.5 and 5.0 apps.

Posted on Wednesday, June 18, 2008 by Alan Telford

Great article. However, you don't have to manually tweak the export file. You just need to set the following property "Global Properties | Global Objects | Don't generate globals" to TRUE, in your ABC.APP

This stops the ABC.APP from generating and exporting GlobalRequest (and other) fields into the EXP file.

For example, I just tested on the PEOPLE.APP in the clarion examples folder, and created the ABC.APP and DATAROOT.APP using the following settings:

```
ABC.APP (GLobal Properties)
GENERAL
  Generate template globals and ABC's as external   NO
GLOBAL OBJECTS
  Don't generate globals                            YES

DATAROOT.app  (GLobal Properties)
GENERAL
  Generate template globals and ABC's as external   NO
FILE CONTROL
  Generate all file declarations                    YES
  Export all file declarations                      YES
CLASSSES - ABC LIBRARY FILES
  Override Defults                                  YES
  Location of ABC library                           ExtDLL
  External Library base name                        abc

PEOPLE.app (GLobal Properties)
GENERAL
  Generate template globals and ABC's as external   YES
FILE CONTROL
  All files are declared in another .APP            YES
```

CLASSSES - ABC LIBRARY FILES

Override Defults	YES
Location of ABC library	ExtDLL
External Library base name	abc

Give this a try. You should find it works well.
And if you're using this "Don't generate globals" in ABC.APP then you don't need to add it as an external dll, as this is handled automatically by the template. The ABC.APP shouldn't show in the "External Module Options" tab. If it does, then you need to uncheck "Standard Clarion LIB/DLL" as it no longer has any global objects such as error class.

Posted on Wednesday, June 18, 2008 by Benjamin Dell

Thanks Guys for the feedback. There are many ways to do this from feedback from such a community as this one. (Clarion)

Both ways of eliminating the global variables seems to work very well...My way was the fastest way for me to do it, one that i know worked, and does not require me to write any extra templates, will work with new Clarion Versions, etc as it is done very very infrequently.

Alan...I have run into problems with some globals not being there when I have global classes and variables exported from custom dlls that are included via templates into my ABC dll....

I will look at this in more detail.

Thanks

Posted on Wednesday, June 18, 2008 by James Palmer

Ben:

Thanks for the great article. One question: Would this (or a variation thereof) be a way to introduce multiple dictionaries into a system? It seems to me that it was the duplication of the ABC classes that always stood in the way.

Posted on Wednesday, June 18, 2008 by Benjamin Dell

Hi..as an introduction to multi dictionaries....

Yes. If you are prepared to use hand coding to access these files and file classes without using template prompts. There is also a freeware template available that allows you to import the exported procedures from a second or 3 rd dictionary dll into your application.

I have not used it myself as I tend to include the tables, screens, reports and source procedures that work with certain tables to provide functionality (like security, unit of measure conversions, geneology, stock control, etc) into completely self contained function libraries (dlls) that can be used by any of my applications.

This basically boiles down to about 6 different 'dictionaries' being used in the same program, even though the tables in the other 'dictionaries' are not available to the appgen.

Posted on Wednesday, June 18, 2008 by Rhys Daniell

Anybody know how to make this work for us Legacy dinosaurs? When using the IDE in non-ABC mode, some of those template prompts aren't available.

Posted on Thursday, June 19, 2008 by Carl Barnes

"how to make this work for us Legacy dinosaurs?"

There are 2 legacy issues I can think of.

#1 the simplest problem is if you desire to make a library of functions as a DLL then having the $GlobalRexxx exports cause a problem when linking it in. The article method of editing the LIB or my suggestion of editing the EXP will fix that problem.

#2 the App_RD App_UD App_SF module functions (RI Delete, update and standard functions) that are generated into every DLL.

The only part of these that could be considered similar to ABC are the SF module "standard functions". They are very small (40K of source) compared to ABC (9MB of source). But they exist in every DLL. It is pretty easy to modify the templates to optionally: 1) generate those functions into only one DLL and export, 2) have them as external in other DLLs.

IMO the RD/RU modules are the real problem with legacy templates. They generate file RI code into every DLL for the files used in that DLL. Some 3rd party has a product to modify the templates to generate the RI code for all files into one data DLL and use as External from all other DLLs. I think Buttercup (the infamous Tom

Mosley) did it originally and it was named DET the Dictionary Enhancement Toolkit. It got sold to Mitten.

Add a comment

Clarion Magazine

Creating Directories With #SERVICE

by Dave Harms

Published 2008-06-18

I'm in the process of writing a template chain to create Clarion# Model-View-Controller web applications based on Castle's Monorail, and as a result I've come to appreciate the little-known #SERVICE template statement.

Little is known about #SERVICE because little is said. Here's the help entry:

> The #SERVICE statement is for internal use only. It is not documented and is listed here simply because it is visible in the shipping templates.

#SERVICE appears to be a specialized version of #RUNDLL.Here's one usage from the shipping templates:

```
#IF (VAREXISTS(%Host32) AND %Host32)
 #SERVICE('C60TPLSX.DLL','GenReadABCFiles')
#ELSE
 #SERVICE('C60TPLS.DLL','GenReadABCFiles')
#ENDIF
```

There are a variety of DLLs called by #SERVICE, all of which begin with C and have TPL in the name. The second parameter of #SERVICE is one of the following:

- GenReadABCFiles
- GenViewABCClasses
- GenGetColorSuffix
- GenFileService
- GenOLEDBService
- GenExpEdit
- GenGlobalExpEdit

Each of these are function calls; the one that caught my eye is GenFileService.

Creating directories with #SERVICE

I came across #SERVICE and GenFileService by accident. My template has to create certain subdirectories if they don't exist, and at first I used #RUN with a MKDIR command. But that didn't work reliably, at least not on Vista. Since my templates really don't need to create directories that often, I resorted to writing a batch file with the templates and executing the batch file manually as needed.

And then one day I searched my template directory for 'MKDIR', looking for my own code and came across this #GROUP:

```
#GROUP(%ValidateDirectory,%pTempDir)
```

```
#IF(~EXISTS(%pTempDir))
   #SERVICE('C60TPLS.DLL','GenFileService', 'MKDIR ' & %pTempDir)
#ENDIF
```

This is exactly the functionality I'd been looking for - a #GROUP (a.k.a. a template language function) to create a missing directory. I plugged it in to my template and it worked perfectly:

```
#INSERT(%ValidateDirectory,'.\Controllers')
#INSERT(%ValidateDirectory,'.\Views')
#INSERT(%ValidateDirectory,'.\Views\Layouts')
#INSERT(%ValidateDirectory,'.\Views\Rescues')
#INSERT(%ValidateDirectory,'.\Models')
```

Note that this #SERVICE call will not create multiple subdirectories at once. In other words if .\Views doesn't exist I have to create .\Views\ before I can create .\Views\Layouts.

GenFileService also supports RMDIR and CHDIR commands. Keep in mind that if you use CHDIR you are changing the directory for the Clarion environment; you'll probably want to save your current directory first and restore it as appropriate.

I haven't explored the other #SERVICE options - if you have please let me know how you've found them useful. And as always with undocumented features, use #SERVICE at your own risk. And keep in mind that Russ Eggen was once told by the London development team that #SERVICE is a dangerous command. That said, I think the specific usage I've demonstrated should be fairly safe.

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

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

More Aussie DevCon commentary

Direct link

Posted Thursday, June 05, 2008 by Dave Harms

Russ Eggen passed along some comments from Aussie DevCon attendee Kim Davies, with Kim's permission to publish those comments in the mag:

> Best one I've been to. The support from 3rd Party developers was astounding with prizes and gifts and discounts. The SV guys did a great job and really boosted the confidence of everyone attending with their demo's and training. Very enlightening.
>
> The ability to use Clarion7 Win32 mixed with .Net and vice-versa was something to see. Not straight forward yet, but I assume, eventually it will be.
>
> The issue now, will be for most 3rd Party Developers to reposition themselves in light of the .Net product. Even though Claion7 Win32/64 whatever, will continue to have a lifecycle of it's own, I belive the majority will eventually migrate to the .Net platform. And competition in the component world is steep.
>
> I must admit, I felt that the appgen was pretty impressive, but they said most of the problems relate to the conversion process.

Kim's sentiments echo what I've been hearing from other attendees. Although AppGen wasn't made available there clearly were a lot of positives to the event.

As I've said before, .NET brings both challenges and opportunities for Clarion third party vendors. Yes, there's a lot of .NET code out there and some of that readily replaces existing third party products. On the other hand if you create a third party product for Clarion#, and it isn't tied to something that's uniquely Clarion (such as the template language or the file driver system), your potential market is suddenly not just Clarion developers but *all* .NET developers.

The day the data center exploded: lessons still being learned

Direct link

Posted Thursday, June 05, 2008 by Dave Harms

As you probably know by now Clarion Magazine was affected by a major electrical explosion at ThePlanet's Houston H1 data center. Some 9000 servers (and close to 3/4 of a million web sites) were affected. Although 6000 of those servers were brought back online within a few days, Clarion Magazine's primary server is still down. Clarion Magazine has been running on the backup server since Sunday, although with delays in DNS propagation not everyone may have been able to access the new site that quickly.

Unfortunately, email has been a different story. In part that's due to what I see as a design flaw in the data center, and in part it's due to bad planning on my part.

When the electrical room blew it took out not only 9000 servers but also the DNS servers supporting all those machines. That I consider a design flaw - a complete failure of the data center should not also have taken down DNS. I keep my domain registrations with a separate company (I use both DirectNic and GoDaddy) so I wasn't completely reliant on ThePlanet and was able to get the backup server online via GoDaddy's DNS servers. After a few days ThePlanet got full DNS administration restored, so I've gone back to my usual DNS admin setup.

But email is trickier, mainly because of spammers. One of the ways to filter out spammers is to do a reverse lookup on the domain from which the email is coming. If the name and the IP address don't match up the mail gets dumped. ThePlanet, like other ISPs, lets me set up the reverse lookup so this doesn't happen.

Now on to my bad planning. When I set up this particular backup server last year I migrated the email settings, but I had one problem and I missed one task. The problem was that I somehow munged the clarionmagazine.com site (which I use for the mag's email) in Ensim and couldn't delete it (I don't actually use Ensim to admin the site, just to admin users and email). I didn't follow through and fix that problem. I also never set up the full DNS on the backup server.

So cut to early this week. I've moved the web site, and I'm waiting on news that my server is coming back online. Given the hassles with DNS changes it would be best if I could just get my original server back within a day or two. And the news out of Houston seemed positive, despite a major setback on Monday with a faulty backup generator. Still, just for some insurance I leased a third server (from a different company) on Tuesday; more on that another time, but safe to say it's taking longer than anticipated to bring that one online as well.

Now it's Thursday and I finally got through to a tech who admitted it could be another day or two before all the servers are back online. I'll give him the benefit of the doubt and say he's probably being optimistic; I'm no longer holding my breath.

I did manage to fix the clarionmagazine.com Ensim problem on the backup server (no excuse for not persevering on this last year). And belatedly I requested the domains be added to the new server so I can administer DNS and get my reverse DNS lookup configured, but of course domain additions only happen once per day and I guess it'll be tomorrow before I can update the DNS for the new server. Meanwhile I'm using the DNS for the old server to point to the new box. That lets you send me email (once you get the DNS changes) and it lets me send you mail, but there's a decent chance your server will dump it because the reverse DNS lookup doesn't match. Hopefully that bit will be fixed by tomorrow.

The only thing I can't easily migrate right now is the ClarionMag news server (the NNTP application that is, not the physical server). My backup there is a bit outdated, and in any case my experience with moving the newsgroups from one machine to another is that the only way to do it and not mess up everyone's news readers is to shut down the news server, take a backup, and restore the backup to the new machine. So that task will have to wait until the primary server is back online (or else I'll have to start the newsgroups up from scratch).

It's been an adventure, and I'm grateful the consequences weren't worse. I expect to post a more complete article on my misadventures in disaster recovery in the near future. Meanwhile, thank you again for your patience.

Whiners vs conference attendees: And the winner is...

Direct link

Posted Wednesday, June 04, 2008 by Dave Harms

It's been interesting to watch the discussions that have developed in the wake of the Aussie DevCon. It seems to be the case that anytime there's a beta release of C7/Clarion# or an announcement of some new feature there's a fresh round of cynicism about the time it's taking to get a gold release of either product (although most of the really violent sentiment seems to center on C7). The Aussie DevCon has been no exception: although AppGen was shown to attendees, there was no release nor was a firm date given for that release (although there was some hint that third party vendors would soon get AppGen for compatibility testing - time will tell).

To be fair to the complainers, some of that cynicism is justified. It really has taken a lot longer than anyone wanted or expected to get even the current AppGen-less beta. But the attitude of the critics who weren't at the Aussie DevCon is in stark contrast to that of the attendees, at least the ones I've heard from who came away with fresh interest in Clarion and hope for the future.

I think there are several reasons for this disparity. One is that SoftVelocity doesn't communicate that much with its customer base, and darkness fosters despair. The folks in Eden got a little more light last weekend than the rest of us have had for some time.

It's also possible that the attendees felt better about Clarion simply because they'd paid to come to a conference and they couldn't afford not to have a good feeling about things. This is along the lines of the idea that you enjoy a wine more if you know it's expensive. I'm not sure that's strictly true, however. I vividly remember the despair that set in at the 1999 Florida DevCon when it appeared that Clarion would be gutted in TopSpeed's rush to DotCom riches.

A slight variation on this theme is the idea that those going to the conference predisposed to feeling good about whatever SoftVelocity is doing.

Maybe.

Or maybe there really is some good news here. On the C7 front SV did, after all, actually demonstrate an AppGen (if somewhat carefully). And that's the last major piece of the C7 puzzle. The new AppGen is capable of converting C6 applications. Testing continues. Some apps convert correct, others require further bug fixes. It's all important progress.

As I said most (thought not all) of the whining and complaining is about C7, not Clarion#. I've even seen a post suggesting that it could take a year for third party support to line up behind a C7 gold release. All I can say to that is if your business plan depeneds on vendors who take that long to respond to major release of Clarion then you have way bigger problems than the C7 delay.

I strongly suspect that the upbeat attitude coming out of the Aussie DevConhas at least as much to do with the Clarion# sessions as it does with the C7 AppGen. It's a long, long time since Clarion programmers had much to be excited about in the way of new language features. The first Clarion for Windows release comes to mind: that opened up a whole new world. Certainly object-oriented support and the ABC class libraries were big changes as well, but in terms of how OOP and ABC impacted the kinds of applications Clarion developers can create I don't think they compare with the transformation of Clarion into Clarion#.

When Clarion 4 introduced OOP and ABC, Bruce Barrington said "Look, it isn't your daddy's Clarion anymore." To paraphrase BB, it isn't your grand-daddy's Clarion anymore. Yes, C7 is the vital next step, but C7 isn't the future. Clarion# is the future.