Clarion Magazine

This edition includes all articles, news items and blog posts from September 1 2011 to September 30 2011.

Clarion News

Read 7 Clarion news items.

The ClarionMag Blog

Read 2 blog entries.

Articles

Tip of the Week: The Dark Side of Omit()

September 2 2011

Omit() is a handy way to exclude source code from compilation, both conditionally and unconditionally. But be wary of falling into these Omit() traps.

Tip of the Week: Do You Have Overgrown Embeds?

September 9 2011

Do you have too many embeds in your procedures? Do you have too much code in your embed points? Here's a rule of thumb that may help.

Tip of the Week: We've All Done It, And Here's Why We Shouldn't

September 16 2011

Here's one feature that really doesn't have any business being part of the Clarion language.

Tip of the Week: Optional Return Values

September 23 2011

Sometimes you don't need a return value from a certain procedure or method. But not receiving one leaves you with a compiler warning.

CIDC: Monday Keynote

September 25 2011

Notes on Robert Zaunere's keynote address.

CIDC Reception

September 25 2011

CIDC 2011: Notes on the Sunday evening reception.

CIDC Day One Notes: Writing Code Backwards

September 25 2011

CIDC Day One Notes: Clarion And The Integrated IDE

September 25 2011

CIDC Day One Notes: Shawn Mason

September 26 2011

CIDC Day One Notes: Bruce Johnson

September 26 2011

CIDC Day One Notes: Fun With CapeSoft

September 26 2011

CIDC Day Two Notes: Mike Hanson, Visual Design Tips and Tricks

September 27 2011

CIDC Day Two Notes: Application Generation in Clarion.NET

September 27 2011

CIDC Day Two Notes: Bob Foreman - Big Data

September 27 2011

CIDC Day Two Notes: Mark Goldberg on APIs

September 27 2011

CIDC Day Two Notes: Andy Wilton, Using COM

September 27 2011

CIDC Day Two Notes: Andy Wilton, Noyantis Tricks

September 27 2011

CIDC Day Three Notes: Using Clarion For Mobile and Web Integration

September 28 2011

CIDC Day Three Notes: Thin@

September 28 2011

CIDC Day Three Notes: Panel Discussion

September 28 2011

Clarion News

Glossy Buttons Pack

The Glossy Buttons Pack has been released. This new pack includes the Glossy Buttons Themes, which were designed for round buttons without any text. The action of the buttons is suggested just by an icon. Icons are included for the standard buttons (add, modify, delete, view, select, accept, exit, cancel, help, etc.) in the same hue for each topic. Demo available. This pack is available for downloading for all customers with an active subscription program.

Posted September 16 2011 (permanent link)

New C8 Gold installs for CPCS, RPM, AFE & Intelligent Mail Barcodes

New C8 Gold installs for CPCS, RPM, AFE & Intelligent Mail Barcodes are now available for download. As indicated in the C8 Gold announcement email any tools that ship local link LIBs should be rebuilt in version 8. To that end the libraries have been recompiled so there won't be any surprises down the road.

Posted September 16 2011 (permanent link)

EasyFingerPrint

EasyFingerPrint is a set of classes and templates allowing you to use Fingerprint SDK (Griaule Biometrics) 2009 for ActiveX in your Clarion applications. Fingerprint SDK is a software development kit (SDK) that enables your applications to use fingerprint recognition. As a software developer you can integrate biometrics into your software all through a single, intuitive interface. EasyFingerPrint Clarion template: automatically generates Initialize/Finalize of ActiveX control in your Clarion procedure; automatically displays acquired image in the IMAGE control; enrolls; performs a verification by comparing the two supplied templates (current and stored in the database table); saves the template into the DB; displays current enrolling quality in PROGRESS control. EasyFingerPrint requires Clarion C6.3, C7.3 and C8.0, ABC or Legacy; Fingerprint SDK (Griaule Biometrics) 2009 for ActiveX installed (purchased separately). This package includes Classes (full source code), Templates, example and documentation. Price: $99 (New Subscription: 1 Developer license + 1-year Maintenance Plan).

Posted September 16 2011 (permanent link)

FullRecord 3.03

FullRecord is a professional audit trail system. The full record of everything that is changed in the system allows you to know exactly who, how, when and what happened at any time with each record of all your files, and among other things you can recover lost information, analyze operation sequences to optimize the system work, find operational errors and educate the user to improve the use of the system. FullRecord 3.x is fully automatic. Every insert, delete, or change in your system will be audited, even if it is hand coded (No additional code needed on your part). In other words, all your ADD, PUT and DELETEs will be audited automatically (you add no code!) either Legacy or ABC code.

Posted September 16 2011 (permanent link)

amazingGUI for Clarion8

The public release of amazingGUI for Clarion 8 is now available for download. The installation keys will be sent to every customer with an active maintenance plan.

Posted September 16 2011 (permanent link)

EasyCOM2INC 2.13

EasyCOM2INC 2.13 is now available. The EasyCOM2INC utility is used to automatically creating Clarion include files with the definitions of COM-interfaces from IDL file and generate needed Classes. Changes include: Template fix for Clarion 7.0 and above - coinitex.LIB static library no longer needed (CoInitializeEx api function was included in win32.lib); Fix where [out] parameters could not return a value to caller; added a new "Local routines" embed point for the event functions handle. This release is available, free of charge, to all customers who have an active maintenance and support subscription plan.

Posted September 16 2011 (permanent link)

Icetips Update and new Solo Subscriptions

Icetips announces some major changes to the website as well as in subscriptions along with the re-release of the Xplore Browse Templates. Also note the new solo subscriptions. Arnor/Icetips will be at CIDC 2011.

Posted September 16 2011 (permanent link)

The ClarionMag Blog

Monday keynote is online

You can view Bob Zaunere's Monday keynote here. This is a public link.

Posted September 26 2011 (permanent link)

More DevCon info to come

I still have a backlog of notes and photos to get through, as well as commentary on various issues. The bulk of that material will be up on the ClarionMag site next week. I'll do what I can this week although I'm taking a couple of days off with my family who are here with me in Orlando.

Posted September 29 2011 (permanent link)

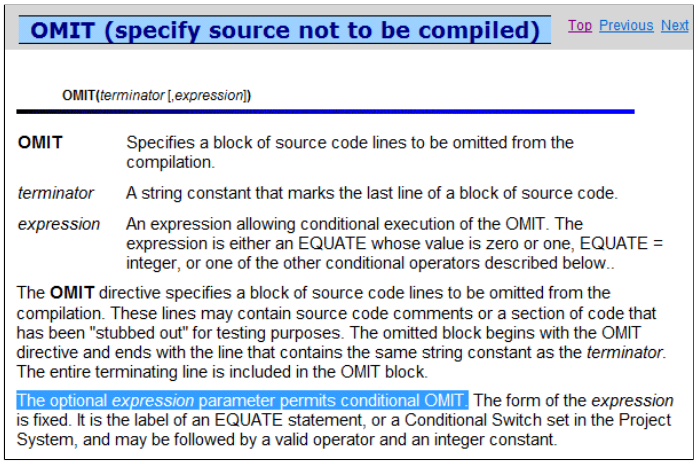
Tip of the Week: The Dark Side of Omit()

By Dave Harms

Posted September 2 2011

What do you do when you have code in your app that you no longer need, but you want to keep around for historical purposes, or perhaps because you're not completely sure that you won't one day need it again? You have two options: comment the code out or `Omit()` the code.

Here's a portion of what the help has to say about Omit:



OMIT (specify source not to be compiled) Top Previous Next

OMIT(*terminator* [,*expression*])

OMIT	Specifies a block of source code lines to be omitted from the compilation.
terminator	A string constant that marks the last line of a block of source code.
expression	An expression allowing conditional execution of the OMIT. The expression is either an EQUATE whose value is zero or one, EQUATE = integer, or one of the other conditional operators described below..

The **OMIT** directive specifies a block of source code lines to be omitted from the compilation. These lines may contain source code comments or a section of code that has been "stubbed out" for testing purposes. The omitted block begins with the OMIT directive and ends with the line that contains the same string constant as the *terminator*. The entire terminating line is included in the OMIT block.

The optional *expression* parameter permits conditional OMIT. The form of the *expression* is fixed. It is the label of an EQUATE statement, or a Conditional Switch set in the Project System, and may be followed by a valid operator and an integer constant.

Notice the highlighted phrase. I really think this isn't the optional way to use `Omit`: this is the *only* way to use `Omit`. It's a tool to conditionally compile code, and nothing more.

Using `Omit` without an expression to always exclude a small block of code works fine:

```
omit('EndOfOmit')

WrongObject.WrongMethod('Really bad idea')
EndOfOmit
```

But if you use `Omit` to exclude a larger block of code, you're making your code much less readable. Imagine that you have a block of code that's too big to fit on screen, and your `Omit` statement and the closing label are both off screen.

How do you know the code is omitted? You don't. There's absolutely no visual clue. Even with a few lines of code it's more difficult to recognize that the code is omitted.

`Omit` does have one other semi-legitimate use, which is to omit generated code using two embed points. This is risky, as it assumes the code will always be generated the same way. I recall more than a few developers having issues with this when converting from Legacy to ABC; `Omit` statements that

previously affected a block of code now omitted parts of if not entire methods.

The Omit alternative

Unless you're conditionally compiling code, just say no to `Omit`. Instead, highlight the code you want to exclude and press Ctrl-/ and the Clarion editor will comment out the block for you. Not only is every line of code preceded by an obvious `!` character, but the editor colors that code as if it were a comment, so it's dead obvious that the code isn't in use. If you need to uncomment a block, highlight it and press Ctrl-Shift-/.

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

Article comments

by Dag Roger Sundmyhr on September 23 2011 (comment link)

This is actually only a problem in <= Clarion6. In >= Clarion 7 the omitted code is folded and therfore easy to spot

But I agree that using omit instead of commenting out "old code" is wrong. Its much easyer just to comment the code. BTW - commented code blocks are allso folded in Clarion 7 and 8 ;-)

I think OMIT is nice when you nead to optionally omit code for special situations, like using project constants to control the compiler. The most common usage of omit is probably the control of included files

OMIT('EndOfIncludeFile', ThisFileIsIncluded)

ThisFileIsIncluded EQUATE(1)

The Code

!EndOfIncludeFile

- Using this instead of include('TheFile'), once is of course an other discussion and has nothing to do with this article...

---

by Dag Roger Sundmyhr on September 23 2011 (comment link)

The above comment was written by

Ole Morten Heien @ Advisor AS

---

by Koen Tjoa on September 23 2011 (comment link)

What is mentioned in this article makes sense. However I consider it sometimes useful to omit a whole procedure. There is a free template from Lodestare Software available which makes that easy. It makes also easy to see that particular procedure is omitted.

See: http://www.cwaddons.com/freebies/omitproc.html

Regards, Koen Tjoa

---

by Dave Harms on September 23 2011 (comment link)

Ole, it's not always easy to spot in C7/8, especially if the omitted block is very large and you end up being dropped into the middle of it.

Koen, thanks for pointing that out.

---

by Guy Bernard on September 23 2011 (comment link)

I've never used the OMIT() statement until just recently when I added some graphs to an application using RMChart, aided greatly by Al Randall's "Using RMChart with Clarion" article published 2006-06-16 -- THANKS AL!

There are a bunch of parameters defined using the RMDesigner that are used to pass input to the RMChart-dll which draws the chart. In Al's example he included the RMDesigner parameters in an OMIT() segment and I followed his lead. It makes the parameter-documentation easily available. Sticking it at the end of the embeded-code segment that contains the RMChart stuff makes it handy and easy to remove-and-replace with copy-paste when RMDesigner changes are made.

I've always commented-out code, and documentation notes, and now I'll not change my style. It's neat to benefit from the experience of others, thanks for the article. Guy Bernard

BACK TO TOP

Tip of the Week: Do You Have Overgrown Embeds?

By Dave Harms

Posted September 9 2011

If you've been reading Clarion Magazine for a while, you've probably got the idea that I'm not big on embeds. Well, actually I'm big on the concept of being able to slot in my code anywhere I want; I'm just not big on putting big chunks of code in embeds, especially if that code contains any business logic (which is often the case).

How to properly move embed code into a procedure or a class is a topic almost as large as programming itself, so I won't even begin to go into it here.

But think about this: How many procedures do you have where you can't see all of the embed points you've used, on screen, at one time? Even with that huge monitor? Or, how many embed points do you have in your apps where you can't see all of the code for that one embed at once?

Code makes sense when you see it in context, which is very difficult to do when you're working in an embed point. The embeditor definitely helps, but it also shows you all the possible embed points and the code that will only be generated if you put code in those embed points. While a solitary embed point gives you no context, the embeditor gives you too much context.

As wonderful as the Clarion template system is for embedding code, it's far from an ideal system for writing and maintaining code. Can you glance at a procedure's embeds and quickly understand what your embed code is doing? If not, it's probably time to start thinking about how you can move some of that code into source procedures and/or classes.

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

Article comments

by Russell Eggen on September 23 2011 (comment link)

Document your code! There is no such thing as self documenting code. Well, yes there is as far as what the code is doing, but rarely why. Why did the developer put this code here? Well, its obvious today as I'm writing it, but what about 6 months from now? Or longer?

So what does this have to do with embeds? Plenty! Document why you need something. Got an app from a client once. Had a procedure that called about 8 reports procedures in a row from code templates. Can't tell from the procedure names which report and why. Simple solution: Add a one line comment source embed before each call.

Now you have a quick one view of a group of embeds that you now know exactly what its doing from the embed tree and the embeditor.

Not all embeds are source! ;-)

---

by Dave Harms on September 23 2011 (comment link)

Thanks Russ, it's always better to be able to read your code. And putting one line descriptions in the first line of each embed is a great technique.

I think the larger question, though, is whether that code should be in an embed in the first place...

BACK TO TOP

Tip of the Week: We've All Done It, And Here's Why We Shouldn't

By Dave Harms

Posted September 16 2011

If I could vote for just one thing to be *removed* from the Clarion language, it would be implicit variables. These are variables you declare in-line, rather than in the data section of a procedure, method or routine. As the help explains:

> The Clarion language provides three types of implicit variables:
>
> - A label terminated by a # names an implicit LONG.
> - A label terminated by a $ names an implicit REAL.
> - A label terminated by a " names an implicit STRING(32).

There are several problems with implicits. One is spelling; if you use short variables like `x#` and `i#` you're probably okay, but if you use `counter#` and then misspell it once as `countter#` you will have two variables, not one. And your code will probably break.

Another big problem with implicits is scope, particularly in ABC. Many embed points in ABC applications are actually inside virtual methods; embed some code, and the AppGen generates a method with that code; the runtime library automatically calls your virtual method at the appropriate time, say for window intialization or browse filtering. Each method has its own scope, however. An implicit variable declared in one method (one embed) won't be visible to any other method (embed). You have to declare a variable either as a class property or at the procedure level for it to be shared between embed points that are created as virtual methods of a class.

If you declare an explicit declaration in one virtual method embed's data section and then try to reference the variable in another virtual method embed, the compiler will tell you pretty quickly that you've made a mistake. Use implicits and you'll be scratching your head, wondering why your code doesn't work.

Implicits may be easy, but they make for sloppy code. As with many things in coding, a bit of extra work up front declaring an explicit variable will pay dividends.

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

Article comments

by Stephen Bottomley on September 22 2011 (comment link)

I use implicits mostly the short i# j# ones but only within the scope of an IF or a LOOP. Once I'm past the statements END I consider it dead and always re-initialse it for use within any other IF/LOOP. Any others I declare.

Steve B.

---

by Dag Roger Sundmyhr on September 23 2011 (comment link)

I do NOT agree with this article!! First of all, implicits follows the exact same rules as defined variables. The documentation is wery clear on that. Just becuase its easy to miss spell a implicit is not an argument to remove the feature. Its nearly the same as saying "Remove references. Its going to crash the program if you forget to NEW the variable"

Saying that this is espessially a problem in ABC is allso a false statement. First of all, its not the ABC that is the problem here, but the usage of implicits in OOP. Every method is actually a procedure with its own scope, and therfore the implicits are considered as locals for that method, but this is allso wery well documented, and frankly, as a proffesional developer, this is an expected behaviour.

The chance for problems because of miss spelling are allso wery well documented in the docs.

I found implicits to be a wery nice feature and uses it in some spesial situations. I do not see any problems with this: loop i# = 1 to records(SomeQueue) Get(SomeQueue, i#) ... end

Using locals for this kind of code would result in heavy use of the embeds because the local data area is not avilable in the embeditor unless you use the Embed Editor (or was it the opposite?!? ;-)

I find myself using implisits only inside the embeditor, which is wery seldom because I normally write classes in the real code editor. And there I never use implicits, because the data area are wery near in the editor..

I think that using implicits to hold important data is a miss use of the feature, and not to be considered as a language problem. Unfortunatly Clarion is used by wery many "templates only users". With this I mean persons that run the wizards and have no idea of how the code behind works at all, and have nearly no embeds at all. No offence!!... Removing a feature just to make sure such "developers" dont make a misstake is in my opinion a backstep of the language.

Many years ago, if I recall correct, Bruce Barrington said in the Clarion documentation (I beleive it was in the DOS edition) that he wanted a language that had all the nice features that could be found in Basic, Pascal and C. He created Clarion to have all these nice features. He's idea was, I beleive, to have a language where everything is spossible, as in "If you nead to do something, just do it". I think Implicits is one of this features.

Regards Ole Morten Heien Advisor AS

---

by Mike Hanson on September 23 2011 (comment link)

Ole, implicit variables offer one very small benefit: you can be lazy and not define them in the appropriately scoped data section. All other aspects of implicits are inherently dangerous. In addition to Dave's comment above, there's also the issue of using a S" without remembering that it's a STRING(32).

The compiler cannot check logical errors, but it's very good at checking your spelling. If you define your variables in the data section, then the compiler has a dictionary to check your work.

If you like living dangerously and enjoy saying "D'Oh!" then use implicits. Otherwise, I suggest you:

- Wear your seat belt
- Don't talk on your cell phone while driving
- Don't run by the pool
- Leave the guard on your table saw
- Don't leave a loaded gun around small children
- And DON'T USE IMPLICITS!!!

---

by Russell Eggen on September 23 2011 (comment link)

DAB told me years ago that from the compiler's viewpoint, implicits are dirty. He implied as far as setting stuff up goes implicits are up there with setting things up.

I don't use them at all, not even in counters. I declare a variable that is AUTO (see past articles where DAB dissects code submissions for why) for counters in a LOOP.

So says the man who used nothing but implicit REALs to code his own APR function :-D

---

by Dave Harms on September 23 2011 (comment link)

Steve/Ole,

I've also used single character implicits in very restricted situations in the past. It's as close to being safe with implicits as you can get. But I've given that up.

> The documentation is wery clear on that. Just becuase its easy to miss spell a implicit is not an argument to remove the feature. Its nearly the same as saying "Remove references. Its going to crash the program if you forget to NEW the variable"

Ole, quite a different argument I think. First, using references generally requires some knowledge of the concepts in play and the risks involved. Second, you will actually get a crash or a compiler error; a null reference is, in my experience, never a silent error. And third, this isn't about whether a feature can cause a crash, it's about whether the problem in the code has visibility - you will never get a compiler or runtime warning. A better analogy is saying Clarion shouldn't allow you to &= (some address) because there's no type checking. And there's still a better reason for that feature because in rare situations there's absolutely no alternative.

There is always an alternative to implicits. Always.

I stand by the article. I don't think implicits have any place in the Clarion language. Of course they're never going to go away, so the next best thing is to never use them.

---

by Jose de Mello Junior on September 26 2011 (comment link)

The ease of the new IDE 32 almost eliminates the extra work of having to exit the edit point to declare a new variable. Soon I fully agree that this type of variable can only stay for compatibility of older projects.

BACK TO TOP

Tip of the Week: Optional Return Values

By Dave Harms

Posted September 23 2011

Last week I objected to the use of implicit variables in the Clarion language. There is at least one forgivable use of implicits, and dealing with procedures that return a value that you don't always need.

Consider this program:

```
   PROGRAM

                                                MAP
                                                    AProcedureThatDoesSomething(),Long
                                                END


     CODE
     AProcedureThatDoesSomething()

 AProcedureThatDoesSomething                    procedure
     code
     ! Okay, it actualy does nothing
     return 0
```

The compiler will give you a "Calling function as procedure" warning on the call to `AProcedureThatDoesSomething()`.

Warnings are best dealt with, and one way to do that is to store the return value in an implicit variable that you don't use any way (which the very best use of implicits!).

```
result# = AProcedureThatDoesSomething()
```

Implicit variables aren't the only solution, however. You can also wrap the function in an If statement that does nothing other than evaluate the function call:

```
If AProcedureThatDoesSomething().
```

Note the concluding period. This is Clarion's statement terminator, and is a shorthand version of the `End` statement. The above line is a more compact form of

```
If
  AProcedureThatDoesSomething()
End
```

And in fact

```
If AProcedureThatDoesSomething() End
```

works also.

I'm a little inconsistent about periods. I don't mind them on a single line statement because they make the statement look like a sentence, but in a nested structure I think they're too easy to miss:

```
If
  AProcedureThatDoesSomething()
.
```

But the best solution, if you are able to modify the code that declares the procedure, is to add a `Proc` attribute:

```
Map
  AProcedureThatDoesSomething(),Long,Proc
End
```

The `Proc` attribute suppresses the compiler warning. Now you have the choice: you can call the code like this:

```
AProcedureThatDoesSomething()
```

or like this:

```
Result = AProcedureThatDoesSomething()
```

and you no longer get the compile warning.

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

Article comments

BACK TO TOP

CIDC: Monday Keynote

Posted September 25 2011

Any official content for this article will appear as comments during the conference and will be incorporated into the article body after the conference.

All other comments will remain in the content section.

Article comments

by Paul Konyk on September 26 2011 (comment link)

Bob Z. summarized the history of Clarion starting at Clarion for DOS 1.0 and contrasted it with Microsoft's VB history and all of the changes in their technology. On the other hand, MS's data access strategies have been long, varied and most have now been depreciated.

Lesson: Clarion has been consistent over the years. Very little code refactoring has been required.

by Dave Harms on September 26 2011 (comment link)

I'll post a few comments later; for now I'll just note that Z said he would be showing the Clarion.Net AppGen tomorrow.

by Russell Eggen on September 26 2011 (comment link)

Anything else?

by Dave Harms on September 26 2011 (comment link)

It was a pretty low key keynote - no big announcements, more of a state of the union address that set a tone of "while MS regularly pulls out the rug from under its developers, we're still here making sure Clarion code continues to run without major modification." Fair comment - it was a bit disturbing to see, for instance, the very long list of data access technologies MS has gone through over the last 25 years.

Bob's keynote address is now online, and it's public access.

by David Swindon on September 27 2011 (comment link)

Thanks for making this available Dave and Everyone involved.

BACK TO TOP

CIDC Reception

Posted September 25 2011

Any official content for this article will appear as comments during the conference and will be incorporated into the article body after the conference.

All other comments will remain in the content section.

Article comments

BACK TO TOP

CIDC Day One Notes: Writing Code Backwards

Posted September 25 2011

Any official content for this article will appear as comments during the conference and will be incorporated into the article body after the conference.

All other comments will remain in the content section.

Article comments

BACK TO TOP

CIDC Day One Notes: Clarion And The Integrated IDE

Posted September 25 2011

Any official content for this article will appear as comments during the conference and will be incorporated into the article body after the conference.

All other comments will remain in the content section.

Article comments

by Paul Konyk on September 26 2011 (comment link)

Some nice improvements to SVN integration with C8EE. Version management is pretty easy.

BACK TO TOP

CIDC Day One Notes: Shawn Mason

Posted September 26 2011

Any official content for this article will appear as comments during the conference and will be incorporated into the article body after the conference.

All other comments will remain in the content section.

Article comments

by Dave Harms on September 26 2011 (comment link)

Shawn began his session with the assumption that we'd all converted to SQL, and were ready for some tips and techniques.

Do you absolutely have to learn SQL just because your databases? No. Should you? Absolutely. Otherwise you won't be able to have a lot of fun with it.

Benefits:

- instant data query
- data fixes without coding
- speed up/optimize code
- intermingle with other programs

Tips

- use Date for dates, not Longs.
- choose where your RI lives (eventually you will want it on the server)
- just don't have RI in both places
- add a primary key to all browse sort orders
- move appropriate code to Prop:SQL (e.g. mass deletes or sometimes mass updates)

...

---

by Dave Harms on September 26 2011 (comment link)

Processes and reports. Shawn likes back end views for reports. He explained the difference between Clarion views and back end SQL views. A view is nothing but a query.

Inner joins. It's important to understand the difference between inner and outer joins (there are others).

Inner join: give me only parents that have children.

Outer joins: give me every parent and any children. Always use left outer joins. Don't confuse

yourself with right outer joins.

...

---

by Dave Harms on September 26 2011 (comment link)

Be careful about using inner joins.

Replace the dictionary autonumbering with the backend Identity. Backend Identity is more stable, faster.

Shawn prefers Identity fields over GUIDs. Most joins are on primary keys, which use clustered indexes. GUIDs don't always sort last, which causes a mess because the index has to be reordered all the time.

Consider server side triggers and stored procedures.

...

---

by Dave Harms on September 26 2011 (comment link)

Some discussion of how to do parent/child inserts where you need the parent identity - @@identity can potentially return the wrong value, so use scope identity.

If you're going to insert or delete more than 7-10% of your rows, then drop and recreate the indexes with `alter index drop`.

`Prop:SqlFilter` can really speed up browses. Lots of power there.

...

---

by Dave Harms on September 26 2011 (comment link)

SQL Candy

Common table expressions are temporary result sets defined within the execution scope of a single select. Can be used for recursive queries.

You can merge data from two sources using the T-SQL Merge statement.

Indexes - SQL probably doesn't need or want your indexes (except for the primary).

Shawn demonstrated the profiler and a template called Tuning. Run it, and it will start logging everything happening in the system. Let it run during production. Then stop it and use the engine tuning advisor to load up the trace file. Look for the PDS (physical design structures) options. If you have a converted Clarion app choose "do not keep any existing PDS" except for primary keys. The advisor will analyze the log and tell you what it needs and you will still have the options of editing the script.

And please use a real defragger like DiskKeeper or PerfectDisk.

...

---

by Dave Harms on September 26 2011 (comment link)

And a bunch more. If you ever get the chance to take in a SQL session with Shawn you'll learn a ton.

---

by Robert Barton on September 27 2011 (comment link)

Clarion is supposed to be RAD Software.

It's handling of SQL access is anything but RAD. I'm well aware there are many versions of SQL, but surely it's not beyond SV of a 3rd Party developer to supply a massive simplification of how to use SQL within clarion.

The use of prop:sql etc. proves my point - as developers we do not need this! I understand SQL is different but so are all those file systems that clarion already handles so there should be a similar simplified way of accessing SQL files.

I just want to be able to use the current methods with SQL files. We need a best practice template (s) to handle SQL without recourse to prop:sql. Would I pay money for such a simplification, you bet your life I would!

How much I hear you say - well let's start at $500 plus a yearly sub of $200 to follow!

I develop systems - not write code! That's why I use clarion and the 3rd party software. Clarion was a RAD system but it has not moved forward too much in the last few years. The addition of such a template(s) would help restore it's RAD.

---

by Eddie on September 27 2011 (comment link)

Well, truth be told, you don't HAVE to use prop:sql. Prop:filter will work fine with SQL, while still offering much of the speed.

Prop:sql and triggers and other stuff mentioned above is only if you want to max out your SQL experience. But it is quite possible to have a full application that doesn't use any of that.

BACK TO TOP

CIDC Day One Notes: Bruce Johnson

Posted September 26 2011

Any official content for this article will appear as comments during the conference and will be incorporated into the article body after the conference.

All other comments will remain in the content section.

Article comments

by Paul Konyk on September 26 2011 (comment link)

Bruce is wearing trousers. And, claims that he is the most handsome third party developer. So, this is going to be serious. Just kidding.

The Road Less Traveled

What Robert Frost can teach us about programming in the 21st century.

Think about the big picture. Where are me going (as developers) and who are we? A significant number (50%) of developers are not independent developers. A third of those at CIDC are employees and not self employed.

The news out there is of depressing. Lots of rapid change is occurring in the world. We're in a very competitive industry.

Consider Shackelton's Expedition (1914) an amazing story of perseverance and survival. Left from South Georgia Island. Months and months preparing over a year stuck and four months organizing a rescue. Not to mention many unimaginable hardships.

Perspective

You can get so focused on such tiny things. Look at the big picture. Everyone in this room is in the wealthiest part of the whole world. Opportunities are enormous. We are in a "golden" are of software development.

Use the tools you -- use the skills you have. Be productive with what you know. There is a difference between shallow and deep programs, shallow and deep programmers.

We're invested in tools, skills, code and customers. It's difficult to suddenly changes all of our skills and tools. If you're a poor Clarion programmer, you'll probably be a poor .NET programmer. Find ways to leverage the skills of others.

by Paul Konyk on September 26 2011 (comment link)

Older applications often are more more feature rich and command a higher price. Do not lightly threw away your existing customer base. We often alienate our customer base. For example, we

are going to re-write our app. It will take five years and it will be prettier. What customer will agree to that?

Impress the Right People

Who are the right people? Our customers. Software should be easier to use. Work at making it better. Software should be better to look at. It not hard, but it takes time to sort these problems out. Don't write ugly programs. Pay attention to detail, make your program look like the programs you think look good.

The world is filled with more non-technical people than technical. iPads are sold to people who are technologically two years old. The person using your program is not a technical whiz. A best, your customer's won't hate you. (Lots of laughs) Most PC users do not want to use computers.

Remember that niche markets are very profitable. That's also why you should raise your prices.

You are not in the software business. You are in the people business. So, consider how your software will be used. Banks are definitely not in the people business. Who can't wait to phone there banker to discuss something with them. Do you want to speak with your cell phone provider. Again, they are not in the people business.

Back to the iPad. Apple is definitely in the people business. They (Apple) care about how heir device looks and feels to the average person. Most people buy them because... One more iPad, one more happy person.

---

by Paul Konyk on September 26 2011 (comment link)

In-House Applications

First off, do you enjoy working for your boss? Make your team better -- who is your customer? Teams can be very satisfying. They can be great ways to work.

Who drives your in-house application? Your job is to make your user's lives better. Show them what is possible. Make your program look good. Have good documentation. You can have a blog, a wiki... Interact with your users.

Custom Application Programming

Always looking for a new client. Do you customers pay a regular maintenance fee? Do they pay you for regular updates? Be clear on your billing and be careful that you don't loose money. All changes need to cost.

Mine them for new work. Do you talk with your customer's at least monthly. The cost yo you is nothing and you are the first to come to mind when some customer work comes along.

Are you investigating similar customers. Do you advertise on your web site. Do you have a blog. Be creative, be profitable. Work on your marketing. Tell people what you do.

Commercial Application Development

Are you communicating with your customers? Do they know your road map? When is the next paid-for upgrade? Will there be new features? What are they. Give your customers a "warm fuzzy feeling". Help you them (customers) feel like they are part of the process.

Analyze any bad experiences that you have had as a customer and make sure that your customers don't feel the same way. Learn from what is happening around you. Try and improve. Keep your

app current. You need to research to determine what can be made better. Build a community of users.

Don't forget to:

GET PAID

Bill early, regularly and stay on top of your billing. Don't start without some cash in hand!

BACK TO TOP

CIDC Day One Notes: Fun With CapeSoft

Posted September 26 2011

Any official content for this article will appear as comments during the conference and will be incorporated into the article body after the conference.

All other comments will remain in the content section.

Article comments

by Dave Harms on September 26 2011 (comment link)

Bruce points out that he will give the iPad away in this session. At the end. Because he's not stupid.

I've had to attend to a couple of things here (and I have a few more left), but I gather Bruce has been talking about CapeSoft's Time and Attendance commercial product.

by Dave Harms on September 26 2011 (comment link)

Bruce has spent a fair bit of time talking about StringTheory. I agree this is a no-brainer. Almost every Clarion app I've seen does a ton of string handling. You can make that code a whole lot more readable and safer with a string class.

Classy of Bruce to note that StringClass was inadvertently based on Rick Martin's string class from ClarionMag. All of which got sorted out a while back.

by Dave Harms on September 26 2011 (comment link)

I bet this is the highest percentage attendance Bruce has ever had for a CapeSoft show and tell. Everybody's waiting on that iPad!!

by Dave Harms on September 26 2011 (comment link)

Bruce sez special pre-release pricing now available on NetTalk 6 (although I don't see anything about this on the CapeSoft web site).

by Dave Harms on September 26 2011 (comment link)

And everyone is playing rock, paper scissors for the iPad.

by Dave Harms on September 26 2011 (comment link)

And Michael Tabakin wins the iPad!

BACK TO TOP

CIDC Day Two Notes: Mike Hanson, Visual Design Tips and Tricks

Posted September 27 2011

Any official content for this article will appear as comments during the conference and will be incorporated into the article body after the conference.

All other comments will remain in the content section.

Article comments

by Dave Harms on September 27 2011 (comment link)

Mike's session focused on two main topics: user interface experience and visual appearance.

You really do need to test your UI. what is obvious to you is not necessarily obvious to your users.

People who say they are expert users usually are not.

The UI should mimic user operations.

Mike had a number of tips for concentrating visual flow and balance, and maintaining simplicity in the UI. Sometimes you can make the user more efficient by making small changes. Again, usability testing is the key.

Use modern fonts! Get rid of MS Sans Serif. Arial is getting pretty old too.

...

by Dave Harms on September 27 2011 (comment link)

Segoe is the present/future.

Clean up your list boxes/browses. Don't pack things in too tightly.

Lots of very specific formatting advice in Mike's presentation - appropriate length for date fields etc.

...

by Dave Harms on September 27 2011 (comment link)

You can use Ctrl-cursor in the designer to move controls to snap lines.

Man, lots and lots of tips.

Mike has done some analysis of icons; he showed a window of different sized buttons with icons, demonstrating which icon sizes will be used and how the text appears. Never make UI elements so they just fit.

The biggest icons in your file should be 128x128 - larger sizes are not supported by all versions of Windows.

You can use all different sizes - you may want icons that go up one pixel at a time.

The alternative to many icons per file is one icon per file but that isn't as flexible an approach.

---

by Dave Harms on September 27 2011 (comment link)

Always have 48x48, 36x36, 24x24 and 16x16. Application icon should be 128x128 and 64x64. Color depth can be 32, 24, 8 and even 4 bits.

There are some good icon editing tools. Mike prefers Axialis (currently on sale at 30% off).

BACK TO TOP

CIDC Day Two Notes: Application Generation in Clarion.NET

Posted September 27 2011

Any official content for this article will appear as comments during the conference and will be incorporated into the article body after the conference.

All other comments will remain in the content section.

Article comments

by Dave Harms on September 27 2011 (comment link)

We are expecting a Clarion.NET AppGen demo today.

by Paul Konyk on September 27 2011 (comment link)

Diego Borojovich started off with Application Generation backed up by Pierre Tremblay and Bob Z.

Diego wants us to "follow the yellow brick road..." -- Appgen. Appgen the only tool that offers two-way code generation. The C6 version of Appgen was over 15 years old. It was stuck at 16-bits and there were problems with 32 and 64-bit OS's

C7 has brought 32-bit IDE and Appgen, .NET and Win32 interoperability. Still some problems with backward compatibility. But, the template language is proprietary and doesn't work with .NET.

C8 has (.NET) has a new Appgen called T4. It uses Microsoft's T4 templates and Softvelocity has extended it's capabilities. T4 is architecturally independent from the IDE. Handles screen design well and works well with Clarion.NET.

T4 (Text Template Transformation Toolkit)

Softvelocity's extensions to T4 include: application and procedure extensions and controls, wizards and embeds. Dictionary information is also available to the T4 templates. A new T4 editor has been written to make writing templates easier. Also included is a template debugger complete with breakpoints, stepping, etc.

Templates can be extended with .NET assemblies. The entire interface is exposed to 3rd party developers to access the IDE/Template services. Appx, the templete files, are written in an open, documented format.

by Dave Harms on September 27 2011 (comment link)

Those extensions to T4 to support embeds are massively important. There's nothing like that in any

T4 usage that I've seen.

---

by Paul Konyk on September 27 2011 (comment link)

SV has extended the T4 template language to provide the capabilities we are used to as Clarion developers. Most of the Win32 functionality has been exposed with the SV T4 extensions.

To write T4 templates, you need to use C#. Templates can be extended with external DLL's. SV assemblies, dictionary and application proprieties are all available to your template. All of .NET's power is at your disposal.

---

by Dave Harms on September 27 2011 (comment link)

In .NET you can run a utility template directly against a dictionary.

---

by Paul Konyk on September 27 2011 (comment link)

T4 editor has code completion, code coloring and debugging.

The SV T4 templates seem very powerful. They can generate a complete .NET application (.cln). SELF.InitializeGenerator() is how the .cln embeded code is interfaced to the T4 generated WinForm and other generated code.

---

by Paul Konyk on September 27 2011 (comment link)

New T4 .cln applications will be different in one significant way. Screens generated by the generator are kept in a separate file from the application logic and embeded code. This is because WinForms should not be messed with or they get broken. This is a .NET peculiarity.

---

by Paul Konyk on September 27 2011 (comment link)

Four weeks after CIDC 2011 ends the Appgen for Clarion.NET will be released. This is directly from Bob Z.

---

by Dave Harms on September 27 2011 (comment link)

I've got a bunch of screen shots I'll post later (hopefully today).

Boy, lots there to digest. Yes, the .NET AppGen approach is quite different from the one we're used to, and for the most part that's a good thing although I think it will be quite an adjustment for many Clarion developers.

What we saw was a wizarded app (although we didn't get to see the app run - need to get Pierre to show that tomorrow). And a lot of explanation of how the new templating system works. The template language bears a lot of similarity (in functionality) to the current language and goes well beyond it in some respects.

There's a major difference in how designers are handled...

---

by Dave Harms on September 27 2011 (comment link)

So in Clarion as we know it, there's a tight coupling between the AppGen and the window designer. In .NET that coupling is less tight. There's good and bad about that.

I have a bunch of thoughts about the new AppGen, but they'll have to wait until later as Mike Hanson's session has started.

by Dave Harms on September 27 2011 (comment link)

Actually one more thing, while I'm keeping an ear on Mike.

I personally think the decoupling of the AppGen and the screen designers is good because it makes code generation independent of and adaptable to individual UI technologies, whether WinForms or XAML or whatever.

But will this make .NET dev easier than, or as easy as, Win32?

IIRC while Clarion for Windows 1.0 made /Windows/ development easier than it was in C, Windows was still a bunch trickier than DOS.

BACK TO TOP

CIDC Day Two Notes: Bob Foreman - Big Data

Posted September 27 2011

Any official content for this article will appear as comments during the conference and will be incorporated into the article body after the conference.

All other comments will remain in the content section.

Article comments

by Dave Harms on September 27 2011 (comment link)

Bob began by mentioning David Bayliss, who also works at Lexis Nexis (and who recently officiated at Bob's son's marriage).

LexisNexis has open sourced its HPCC product, which is a competitor to Hadoop. Hadoop is used by for large scale search capabilities for sites such as Amazon and Facebook.

So what is big data? Taking data from all sorts of sources, both unstructured and structured data, processing the data, cleaning it, joining records together. At the end there's some data that's useful to the customer.

LexisNexis doesn't sell data, they sell information. It's all about processing data.

...

by Dave Harms on September 27 2011 (comment link)

LN deals with petabytes of information.

Why does this matter to Clarion devs? You may not deal with massive amounts of data now, but it could happen soon.

Both HPCC and Hadoop use a "shared-nothing" architecture.

Hadoop has a key-value data model. HPCC supports a flexible data model.

Hadoop uses a MapReduce paradigm, and was designed for search engine applications. HPCC has a more flexible approach.

ECL - enterprise control language. They feel they've solved the declarative language problem. In a declarative language your code describes what you want done, not how you want to do it.

Bob hasn't said it yet, but the C in ECL also originally stood for something else.

...

by Dave Harms on September 27 2011 (comment link)

HPCC has an extract, transform and load (ETL) engine (Thor) that is massively parallel. Thor is programmable using ECL.

Roxie - Rapid Online XML Information Engine. Also programmed using ECL. The "R" originally stood for "Richard" (as in Chapman).

You define the data, you process the data, and then you deliver it to the customer.

...

---

by Dave Harms on September 27 2011 (comment link)

ECL is self-optimizing.

The HPCC team has about a hundred patented algorithms in use. And it's a single consistent language across the entire platform.

Some Clarion people went to Seisint; others went to ChoicePoint. LexisNexis bought both these companies, and that brought a bunch of Clarion folks together again.

David Bayliss wrote ECL in 2000 in response to an Equifax problem; Richard Taylor wrote the first 33 page reference manual.

Equifax had a SQL-based process for predicting who would go bankrupt in the next 30 days, but it took 26 days to run the data. That first ECL implementation solved the same problem in 6 minutes.

...

---

by surfersteve on September 28 2011 (comment link)

at last we see the breath and depth and intellectual fire power, very impressive ! humbling almost

---

by Geoff Robinson on September 29 2011 (comment link)

"Bob hasn't said it yet, but the C in ECL also originally stood for something else."

Dave do you mean "Clarion" or something else?? Please don't leave it up in the air like that!

Thanks for the reporting for those of us who are not able to be there - the more details the better please.

cheers

Geoff R

BACK TO TOP

CIDC Day Two Notes: Mark Goldberg on APIs

Posted September 27 2011

Any official content for this article will appear as comments during the conference and will be incorporated into the article body after the conference.

All other comments will remain in the content section.

Article comments

by Dave Harms on September 27 2011 (comment link)

Mark began his session with a demonstration of his bricklaying tradesman's software, which uses a number of API calls for features from skinning a listbox to displaying 3D drawings.

What is an API call? It's just a procedure call.

But the docs usually look different. A few things to keep in mind when reading C code:

- #define is usually just an equate
- Struct is the C name for a group
- Hex numbers in C begin with 0x

For common data types see the Clarion help: Programming 3GL Functions in Clarion and Resolving Data Types. There's a lot of useful API information in the help file. One of the common date types is usually prefixed by lp, for long pointer which is a memory address.

...

---

by Dave Harms on September 27 2011 (comment link)

W and A suffixes in the Windows API indicate unicode and ANSI versions of the same API call - use the NAME attribute on the prototype to indicate which version you want.

You can use equates to create Clarion equivalents to C data types. There are some of these built into Clarion.

MAP statements cause two things to happen:

- Builtins.clw gets included in the map
- Win32.lib is linked in

Mark removed a function from win32.lib. A previously successful compile using that call failed, proving that win32.lib is used.

...

---

by Dave Harms on September 27 2011 (comment link)

Mangling, Mark suggested, is like Hungarian notation. Hungarian is for humans, name mangling is for compilers and linkers.

Subclassing is done via the API - this is where you insert your own message handler into the Windows messaging system so you can read all the messages processed by a specific window or control.

You can find a lot of interesting things if you go digging.

...

---

by Dave Harms on September 27 2011 (comment link)

Okay, so it all compiles.

Now it GPFs. Why? Probably something's wrong in the prototype or the data being passed into the procedure. Or perhaps you're missing a RAW or PASCAL attribute. (Carl Barnes points out that a missing PASCAL usually causes a problem on the return.

Not coincidentally, Mark recommends Carl Barnes' Clarion Source Search.

---

by Dave Harms on September 27 2011 (comment link)

Pragmas on class LINK and DLL attributes can also cause GPFs if they're wrong.

BACK TO TOP

CIDC Day Two Notes: Andy Wilton, Using COM

Posted September 27 2011

Any official content for this article will appear as comments during the conference and will be incorporated into the article body after the conference.

All other comments will remain in the content section.

Article comments

by Dave Harms on September 27 2011 (comment link)

Andy began by observing that COM is everywhere in Windows. It is the Component Object Model. It encompasses OLE, OLE Automation, ActiveX, COM+ and DCOM.

Access to any COM object is via a standard interface. The key is to understand the available methods as a tree structure you navigate.

Exercise due diligence when choosing COM controls.

Be aware that the COM library must exist on the user's system, so if a license is required that has to be there too. Also each control must be registered prior to use. If our app does the registration than it has to run with sufficient rights.

Two good products for exploring COM controls:

- iTripoli TypeLib viewer
- OCXi (Dave Bratovich - in development, keep an eye out)

...

by Paul Konyk on September 27 2011 (comment link)

Why use COM in Clarion (and not be frightened by it)?

Some might say, COM, ActiveX can't be used. Stay away from them is a command reaction.

Where is COM used? MS Office, Internet Explorer, Media Player, etc.

COM - Component Object Model. Also known as OLE, OLE Automation, ActiveX, COM+ DCOM...

Each COM control is constructed to a certain standard. It starts with a predefined interface. There is a collection of objects (class), properties (variable) and methods (procedures).

Look for COM controls from reputable authors. Reliability and stability are most important. Should have a rich feature set and are regularly updated. And, finally, the COM control needs to have a reasonable distribution/royalty cost.

There are a number of housekeeping tasks which are necessary for each PC where a COM control is used. It must be delivered to each PC, licenses needs to to be compiled into each COM control. Finally, it needs to be registered on each PC where it will be used.

How to open a COM control.

Helpful utilitiy: iTripoli http://www.itripoli.com/itlv.asp

---

by Paul Konyk on September 27 2011 (comment link)

There may be several interfaces in a COM control. Select the one that works best for you.

Andy then walked us through using Type Library Explorer and a COM Control Text Application he had written to explore the COM control. This how he found and practiced using the methods and properties the control.

First, COM controls need to be started (initialized). Then commands are executed and values are "set" (assigned) and "get" (read).

VideoLAN VLC Player has a COM control. A short underwater video (.avi) was played over and over using this COM control in a Clarion program. It is possible to speed up the playback. I bet other effects are also possible.

---

by Dave Harms on September 27 2011 (comment link)

Andy's covering the use of callback functions to trap OCX events. Earlier today Mark talked about the winevent callback during his API talk.

I bet you could have a full day of sessions just on using the many different kinds of callbacks in Clarion.

---

by Paul Konyk on September 27 2011 (comment link)

So far the demonstration has shown how to send commands and values to a COM control and get back values from it. The next part demonstrated how to receive events back from the COM control. They are called Callback events. We saw the position in the video clip (MediaPlayerPositionChanged) being updated until the end of the video was reached. Then an end of clip event (MediaPlayerEndReached) was received.

Returning a value with an event is also possible. So, in this case, as the position in the video clip is updated, the number of seconds into the clip is also displayed.

End of the VLC control demonstration.

---

by Paul Konyk on September 27 2011 (comment link)

DSOFramer is a free COM control which works with Microsoft Office applications.

Initialize DSOFramer and then call CreateNew('Word.Document'). To make things prettier set the property Menubar to 0 before the CreateNew(...) method.

Look inside MSWord.olb (shipped with MS Office) in TypeLib Viewer to find the MS Office commands that you need. For example, find the default printer, insert some text into the active document, edit the text, etc.

This whole COM thing has a number of requirements. But, once everything is in place it is very

powerful. It's something we all could use to great advantage in our applications.

BACK TO TOP

CIDC Day Two Notes: Andy Wilton, Noyantis Tricks

Posted September 27 2011

Any official content for this article will appear as comments during the conference and will be incorporated into the article body after the conference.

All other comments will remain in the content section.

Article comments

by Dave Harms on September 27 2011 (comment link)

Andy's getting into his demo of Noyantis Clarion add-ons. The product line includes wrapper templates for the CodeJock, nBit and Office controls. Andy also has a number of source-only add-ons.

Apologies for the lack of detailed notes - as usual Andy had a ton of interesting stuff to show.

Paul Sivak won the Kindle!

BACK TO TOP

CIDC Day Three Notes: Using Clarion For Mobile and Web Integration

Posted September 28 2011

Any official content for this article will appear as comments during the conference and will be incorporated into the article body after the conference.

All other comments will remain in the content section.

Article comments

by Dave Harms on September 28 2011 (comment link)

Bob Z started things off with a brief explanation of how Win32/.NET interop works in Clarion#. When you mark a procedure for export from Clarion#, the linker creates a LIB file that you can then use the same way you use other libs in Win32.

Pierre then launched into LINQ and the application of that technology to file access in Clarion.NET's "LINQ to FILE" provider. You can do all the standard kinds of file access, but LINQ makes many tasks much easier.

The provider is as fast as views, and sometimes twice as fast.

The templates generate a data access layer, which is sort of the ABC file management classes for .NET.

Pierre showed the generated code, which led to some questions and discussion about the new data layer, which is very similar to the ABC equivalent classes, with the addition of nice one-line LINQ queries for situations where you want to do the equivalent of a file loaded list box. Regular browses do support paging.

...

by Dave Harms on September 28 2011 (comment link)

Diego used the example of a web service to show how to call .NET methods from a Win32 program. It's a pretty simple process, although it's also restricted to simple data types. You could move more complex data across by serializing into some standardized format like XML and the deserializing on the other end.

Z then took over the podium to explain the available web technologies, WebForms and MVC.

The big advantage of MVC is testability - it's much harder to test WebForms code.

...

by Dave Harms on September 28 2011 (comment link)

MVC's use of JavaScript and CSS requires more knowledge of those technologies, so while you have much more control, there's also a bit more effort involved. SV will support both approaches in its templates.

Choices for supporting mobile devices:

- Do nothing; rely on the mobile browser to sort things out
- Change your markup on the server when you detect mobile devices
- Use third party controls to target specific devices
- Solve the prblem with CSS and progressive rendering
- Duplicate pages and modify for mobile devices

Z then demonstrated a simple WebForm app.

...

BACK TO TOP

CIDC Day Three Notes: Thin@

Posted September 28 2011

Any official content for this article will appear as comments during the conference and will be incorporated into the article body after the conference.

All other comments will remain in the content section.

Article comments

by Dave Harms on September 28 2011 (comment link)

Marko and Daniel's session looked at Windows Server sessions vs Thin@ sessions. Originally they had two talks scheduled but one of their slots went to Bob Foreman, so some of the introductory Thin@ info is missing from this session.

If you've already seen any of Marko and Daniel's presentations, last year in Denver or at ClarionLive, then you'll know that Thin@ is a thin client solution for Clarion, allowing you to put your Clarion application on a server and let users run that app on the server rather than on their local machines. This lets you use Clarion as a software-as-a-service solution.

As was evident from the presentation, Daniel and Marko are highly performance oriented and have done a great deal of work to optimize Thin@'s footprint.

There were a number of questions at the end of the session about issues such as printing from the client (yes you can).

BACK TO TOP

CIDC Day Three Notes: Panel Discussion

Posted September 28 2011

Any official content for this article will appear as comments during the conference and will be incorporated into the article body after the conference.

All other comments will remain in the content section.

Article comments

by Paul Konyk on September 28 2011 (comment link)

A discussion about some issues that may have come up during the conference.

To Mike and Bruce. To what extent to Clarion programmers need to code?

Bruce

- keep improving
- get more familiar with source code
- understand source code

Mike

- wrong to think that you can't be as productive with hand code
- need to learn when to hand code and when not to code by hand
- read books about a variety of programming techniques and ideas

Mike Gorman: What three things would you do to increase the Clarion installed base three fold?

Dave

- we're not Softvelocity's sales people and it's our responsibility
- it's not a secret weapon
- be the best developer you can be

Bruce

- the community has a role to play by welcoming new users
- treat each other with respect
- as developers, we have our own businesses to run

Mike

- post messages on social forums about Clarion being relevant and useful

Bob Roos asked how to get better documentation.

Bruce

- get a subscription to ClarionMag

Dave

- read the reference manual
- bring suggestions for articles to Dave
- read the newsgroups, ClarionLive

Mike

- people learn differently tutorials, manuals, hand holding
- do your best, experiment, ask questions

---

by Paul Konyk on September 28 2011 (comment link)

Tammy Hayes when you need programming help, hire young, out of college students.

Mike

- hard to hire someone and bring them up to speed. Try paired programming (two people coding at the same time).
- don't need a Clarion trained programmer, they can learn it very quickly

Bruce

- Yeah

Dave asked, what as the big needs facing the Clarion community?

Charles difficult to find local developers, especially entry level people

- a variety to responses. Depends on whether or not you are a sole programmer or part of a larger team.

Abe Jimenez customers can't find someone to replace a Clarion programmer

Bruce

- it's our responsibility to educate our customers on how and where to kind **real** programmers
- people are not replaceable and a good programmer can learn Clarion or any other programming language fore the matter
- programmers know how to write systems, not how to write Clarion code

Mike

- a developer uses a variety of tools for the job. There are lots of technologies.

What to do if Clarion is not allowed?

Dave

- we can't always get into every market

Mark Holland, we don't have mind share as Clarion programmers. Most programmer can't sell. Need to develop more younger progammers

Mike

- smaller community now. The tool has changed. Nostalgia doesn't really help.

Ed Why don't we have a pool of people to outsource work?

Bruce

- do it yourself if it's worth doing. We all have an idea of what "they" (others) should do.

Dave Harms summarized that we have some important issues to discuss and lets not stop talking about them.

---

by surfersteve on October 1 2011 (comment link)

Nice to see that hand coding ha finally found its place and oop helps that. of course you can alway create ways to generate your code besides the template system

BACK TO TOP