Clarion Magazine

This edition includes all articles, news items and blog posts from September 1 2010 to September 30 2010.

Clarion News

Read 18 Clarion news items.

The ClarionMag Blog

Read 4 blog entries.

Articles

The future of Clarion.NET

September 7 2010

It's six years since SoftVelocity officially announced Clarion.NET. Clarion developers are still waiting for a .NET AppGen, and SoftVelocity has become unresponsive to questions about the AppGen's status. What does the future hold for Clarion.NET, and what lessons can be drawn from the past?

Subscriptions extended by 30 days

September 14 2010

This has been an extraordinarily busy year here at Clarion Magazine. We've launched two new web sites and as a result our publication schedule has been set back. With the launches out of the way we should be back to normal soon; meanwhile, we've extended all subscriptions active May 1 or later by 30 days.

Just launched: DevRoadmaps.com, your guide to .NET development!

September 15 2010

Whether you're using Clarion.NET or non-Clarion tools, our new DevRoadmaps site is an essential tool for making your way in the vast universe of .NET development.

Customizing the Clarion 7 editor's syntax highlighting

September 22 2010

All John Morter was looking for was a better source editor font. What he discovered was a way to teach the IDE some syntax highlighting tricks.

Compiling Clarion Source To Java Bytecode

September 23 2010

What if you could take Clarion source code just as it is and compile it for Java? Andrew Barnham's open source Clarion2Java project does just that, allowing Clarion code to run on Windows, the Mac OS, Linux, or any operating system supported by Java.

Still Staying Connected On The Road

September 30 2010

In 2004 Ned Reiter wrote an article for Clarion Magazine on connecting to the internet while traveling. Six years later Ned's still on the road, and mobile technology is better than ever.

Clarion News

SoftVelocity releases C7.2 build 7583

SoftVelocity has released Clarion 7.2 build 7583 with fixes to the IDE and the RTL.

Posted September 13 2010 (permanent link)

CLARIONLIVE: Call for Third Party Vendors to show at DevCon FREE Show & Tell and Tables

John and Arnold are also looking for drawing prize donors for online and onsite attendees of ClarionLive DevCon 2010. Please contact them at clarionlive--at--gmail.com. Check out the Participant Page for the Presenters, Attendees, and Third Party donors and show-and-tellers.

Posted September 15 2010 (permanent link)

Combit releases free edition of List & Label development component.

combit has entered the freeware market by releasing the Free Edition of development component List & Label. The fully functional reporting tool offers nearly the complete range of features of the Standard Edition and is now available for download.

Posted September 15 2010 (permanent link)

EasyNaviBar 1.03

EasyNaviBar 1.03 has been released! Updated and new demo applications are available. Changes include: Added; Support for the new controls; MDI Frame support; New demo example for the MDI Frame support (based on the standard SCHOOL example); New methods GetControlText, SetControlBorderStyle, SetControlHeight, SetTextBoxMaxLength, SetTextBoxMultiline, SetTextBoxReadOnly, SetTextBoxWordWrap, SetTextBoxScrollBars, LoadRTF, SaveRTF, SetLabelAutoEllipsis, SetLabelTextAlign, SetLabelImageAlign, SetLabelFlatStyle, SetLinkLabelURL, GetLinkLabelURL and SetPictureSizeMode. This is a free upgrade for all customers who have a current (valid) subscription plan.

Posted September 15 2010 (permanent link)

QView

QView is a queue debugging tool. Add a global templateand examine any queue: Features include: Drill down into referenced queues; Access arrays; Filter on any element of the queue; Search the queue (by regex if PCRE installed); Sort, edit and save queue back into the app; Save a queue to delimited file, xml or clipboard; Restore queue content from previous debugging sessions. QView installs for a 30 day trial period. A single-user license is now $85 and covers usage with one or all of C5, C55, C6 and C7.

Posted September 15 2010 (permanent link)

Icetips Utilities build 1.1.2390

Icetips Utilities build 1.1.2390 is available for download for all customers with a valid Gold Subscription. This build contains complete documentation for the Date Class, File Search Class and INI Class.

Posted September 15 2010 (permanent link)

SoftVelocity releases C7.2 build 7583

SoftVelocity has released Clarion 7.2 build 7583 with fixes to the IDE and the RTL.

Posted September 15 2010 (permanent link)

Last day before ClarionLive! price increase

Wednesday, September 15 is the last day to register for the ClarionLive! DevCon in Denver before the price goes up. If you can't make it to CLDC but want to attend the live streaming presentations, you can register for the on-line CLDC. Save even more by registering for CLDC and Capesoft's World Wide Webshop, which covers NetTalk in an intense two-day session.

Posted September 15 2010 (permanent link)

Move to SQL in One Day or Less - Guaranteed

Mitten is holding a one day workshop on converting to SQL. The workshop will be held on November 1st after the Clarion Live! DevCon 2010 in Denver. This workshop includes the following: One day training; All special templates used; Documentation and training materials; Lunch, snacks and beverages; Up to two hours after workshop remote or telephone support. For more details on the workshop, call Mitten Software at 952-745-4941 or visit the web site.

Posted September 15 2010 (permanent link)

Clarion.NET, Clarion#, Visual Studio, C#, VB.NET... Where do I stand? Where do YOU stand?

Arnor Baldvinnson blogs about the long wait for the .NET AppGen, and his decision to write off his investment in Clarion.NET and move to Visual Studio for .NET development.

Posted September 15 2010 (permanent link)

J-Cal Mini

J-Cal Mini is a template which enhances Clarion's calendar lookup control. Clarion's control works fine, but doesn't offer much in terms of theming, customization, settings, etc. Or at least that was the case. Drop in J-Cal Mini, choose a theme, and recompile.

Posted September 23 2010 (permanent link)

J-Tooltips

J-Tooltips was developed a couple of years ago for our Organize365 product. Basically J-Tooltips transforms your existing tooltips into something a lot more useful.

Posted September 23 2010 (permanent link)

J-Calculator

J-Calculator adds little calculator buttons next to your numeric input fields, which your users can click on to calculate the values in the field.

Posted September 23 2010 (permanent link)

PostgreSQL 9.0

PostgreSQL 9.0 includes built-in, binary replication, and over a dozen other major features including: Hot standby; Streaming replication; In-place upgrades; 64-bit Windows builds; Easy mass permissions management; Anonymous blocks and named parameter calls for stored procedures; New windowing functions and ordered aggregates, and more.

Posted September 23 2010 (permanent link)

Codejock SuitePro Bundle expands, Noyantis to charge for upgrades

Codejock have recently announced that they will soon be releasing a new line of Dashboard products including some powerful Charting and FlowGraph controls. Their original plan was to launch these as a new, separate product - differing from their current SuitePro bundle. They have now decided to include the new controls within their existing SuitePro bundle. To cover the cost of the new additions, they plan on increasing the price of the bundle by $100 for new purchases and $25 for renewals (these prices are before the CLARION 30% discount is applied). As a consequence Noyantis plans to introduce maintenance charges for its templates. You can either choose to stay with your current version and pay nothing, you can pay per upgrade, or you can subscribe to an annual maintenance plan. Single upgrades and lapsed maintenance plan: 50% of retail price; Annual maintenance plan: 20% of retail price. Users on annual maintenance plans will receive all updates to their purchased products. The good news is that if you have purchased our SuitePro bundle, then the wrapper templates for the new line of dashboard products will automatically be included in your purchase. Users who purchase a single upgrade will receive the latest version that is available at that moment in time of the particular product(s) they have upgraded. Codejock plan on launching the first of their dashboard products in early November, and so, from the 1st of November 2010, the price of a single developer licence of the Noyantis SuitePro bundle will increase by $55, taking it from $395 to $450. The new bundle will match the Codejock SuitePro bundle product.

Posted September 23 2010 (permanent link)

Noyantis CommandBars 2.07

Noyantis CommandBars version 2.07 is available. This is only a small update that corrects the Memory leak and also adds a Codejock v13.4.1 version selection. The update can be downloaded using the link contained in your original sales email.

Posted September 23 2010 (permanent link)

vuMail 3.20

vuMail 3.20 is available for download. This is a maintenance release that includes the following: Better management of embedded images from a file; Better management of embedded images from a link; Error checking for paths to attachments and embedded images; Resource info added to the DLL.

Posted September 23 2010 (permanent link)

Clarion 7.2 Build 7600

SoftVelocity has released Clarion 7.2 build 7600. This release contains some fixes for recent regressions.

Posted September 23 2010 (permanent link)

CLARION MAGAZINE
READ. LEARN. SOLVE.

Home | Subscribe | E-Books | News | Blog | Store | My ClarionMag | My Lists | Contact

The ClarionMag Blog

The enigma that is Clarion.NET

It's been a long road to .NET for Clarion developers. First announced in 2004, Clarion.NET went to beta as a hand coder's edition in 2007. We're still waiting for the .NET AppGen.

Information about the new AppGen is almost impossible to get. SoftVelocity generally ignores or deletes newsgroup posts asking for an update, and communicates almost solely via occasional blog posts which do little to calm concerns.

For what light there may be, please read The future of Clarion.NET (subscription required).

Posted September 7 2010 (permanent link)

Disappearing menus still a problem

For a while now the C7 runtime has had an issue with randomly disappearing menus. That is, the menus still show but the text is white on white so you can't actually see what you're choosing. Closing the app and restarting it fixes the problem.

A few builds back SV released a special runtime DLL to help track down the problem, but presumably that was build-specific.

If you're seeing this problem, let SV know.

Posted September 16 2010 (permanent link)

How (not) to get updates from Clarion Magazine

For those of you who use the SoftVelocity newsgroups to get your updates on what's happening in Clarion Magazine, please note that this no longer appears to be a reliable option. I've had a number of ClarionMag update postings deleted from the newsgroups.

If you'd like to receive regular updates by email from Clarion Magazine, please register.

If you're already registered and not receiving emails, check your email address and your mailing list preferences.

You can also follow us via our RSS feeds:

- Articles feed
- News feed
- Blog feed
- Everything feed

Posted September 24 2010 (permanent link)

TPS corruption and Microsoft Security Essentials

If you're encountering TPS file corruption, here's a reminder to check for Microsoft Security Essentials. Read more in the newsgroup archive at clarion-software.

Posted September 24 2010 (permanent link)

CLARION MAGAZINE
READ. LEARN. SOLVE.

Home | Subscribe | E-Books | News | Blog | Store | My ClarionMag | My Lists | Contact

The future of Clarion.NET

By Dave Harms

Posted September 7 2010

Six years ago this month, at the only Florida DevCon of the decade, SoftVelocity's president Bob Zaunere announced Clarion.NET, the company's upcoming flagship product for Microsoft's .NET framework.

A little over three years later, in November of 2007, SoftVelocity released the first hand-coder's edition of Clarion.NET. In 2008 there were, by my count, nine builds of Clarion.NET released. In 2009, five builds. In 2010, so far, two builds.

So what do we actually know about the status of Clarion.NET?

At the Aussie DevCon in the fall of 2009 Bob Zaunere predicted a Clarion.NET AppGen beta release by the end of the year, or the beginning of the first quarter of 2010, with a gold release by the end of Q1.

Perhaps the heady atmosphere of Eden was to blame.

Over that past year there have been at least eight blog posts specifically about Clarion.NET:

- August 20, 2009 - How to pre-compile and publish ASP.NET applications.
- September 4, 2009 - The LINQ to File provider.
- March 2, 2010 - A mention of upcoming weekly or bi-weekly builds.
- April 12, 2010 - Full beta (with AppGen) in "about 5-6 weeks" .
- June 14, 2010 - Some screen shots of the new templates and a somewhat unrevealing screenshot of the IDE showing the application properties pad.
- July 7, 2010 - A walkthrough of the template language, covering pretty much the same template snippet as was shown at the Aussie DevCon in 2009.
- August 11, 2010 - A screen shot of the AppGen showing current work on the integration of the data schema and the embeds.
- August 17, 2010 - The managed IP driver.

It's nice to see an increased blogging rate. Even more valuable, however, is some kind of direct communication between SoftVelocity and its customers on the status of the Clarion.NET AppGen.

The communication level around Clarion 7 has been pretty high. For instance, you can regularly find Bob Zaunere posting in the C7 newsgroup, taking suggestions, offering solutions, and generally being responsive to concerns. Ask a question about Clarion.NET, however, and you're likely to get stony silence. You can see this for yourself in the softvelocity.public.clarionsharp newsgroup, where a number of requests for information have gone unanswered. And yes, Bob Z does read and post in that newsgroup.

Clarion.NET - maintaining radio silence

SoftVelocity's silence on Clarion.NET, blog postings aside, is perplexing and raises many questions, including:

- What *is* the status of the Clarion.NET AppGen?
- How can SoftVelocity's predicted release dates continue to be so wildly inaccurate?
- Does the company have the resources to produce a usable Clarion.NET AppGen?
- Is the focus still on WinForms, ASP.NET and Compact Forms (older techologies) or on WPF, Silverlight, ASP.NET MVC and Windows Phone 7 (current technologies)?
- Is the plan still to use an in-house LINQ library to do data access, or will the templates generate code to use the now widely-accepted Entity Framework and/or NHibernate Object Relational Mapping (ORM) tools?
- How is Clarion#, as a layer on top of C#, any improvement over just using C# or VB.NET?
- Does the standalone Clarion.NET IDE have a future, or is the only way forward to recreate the AppGen as an extension for Visual Studio?
- Once there is a Clarion.NET AppGen, how long will it take before the generated application architecture stabilizes?
- Once there is a Clarion.NET AppGen, how long will it take before the runtime library, file drivers, and other supporting code stabilizes?
- How much will SoftVelocity try to make .NET development look familiar to Clarion coders? Certainly that's the whole point of Clarion#. But in trying to preserve familiarity, will SoftVelocity make compromises in application architecture that end up making no one happy?

Perhaps the biggest question is, how did we get into this mess in the first place?

A little history

The story of Clarion.NET goes back at least as far as 1999, when Bruce Barrington sold the Clarion product line to then-employee Bob Zaunere. That product line included an IDE, the 16-bitness of which was already recognized as a liability. More to the point, porting that code to 32 bits wasn't in the cards. I remember one of the London development team members telling me at the 1999 DevCon that there was a lot of 16 bit code in the IDE that no one quite understood. Another inside source has indicated to me that there wasn't a lot of reusable code outside of the file driver system.

If you were SoftVelocity, looking at an already dated 16 bit IDE in need of a complete rewrite, what would you do?

You'd start rewriting the IDE, that's what you'd do. And you'd probably want to improve the IDE as well. If all you did was rebuild the guts as 32 bit code, and it still walked and talked like the old IDE, who would buy it?

And then if the possibility arose to get the source code to an IDE (SharpDevelop) that looked like Visual Studio, and had .NET development built in, giving you a growth path to the .NET version of your product, wouldn't you jump at the chance? Sure you would.

Hindsight is 20/20, and a few Clarion developers I know have suggested that SoftVelocity would have been much better off creating a 32 bit version of the old IDE and *not* adding a bunch of new stuff. That would have taken far less time than integrating with the SharpDevelop IDE, and would have freed up resources to create Clarion.NET as an add-on or extension for Visual Studio. And that might have been a much better choice. But remember that back when SoftVelocity started working on Clarion.NET Visual Studio didn't have nearly the extensibility it has now, and there was also a lot of resistance among Clarion developers (as seen at the '04 DevCon) to having to buy an additional development tool to do Clarion.NET development.

It's my guess that SharpDevelop as the basis for Clarion 7 and Clarion.NET has not provided anywhere near the benefit SoftVelocity anticipated. I've heard there were issues with bugs in the SharpDevelop code base, and certainly the integration of the AppGen with the IDE has not gone smoothly. An IDE is a complex beast, and the Clarion-specific code isn't exactly trivial either. It's taken far, far longer to get C7 into usable condition than either SoftVelocity or the Clarion community would have liked.

Get me rewrite!

I've heard Bruce Johnson say (more than once) that rewriting software from scratch is always a bad idea. Customers expect the new version to do everything the old version does, so you have to make sure that every feature, every nuance is the same. And it takes an enormous amount of effort to do that.

Porting and rewriting are two different things. Porting is where you take the same code you had before and you tweak it for a new platform. It involves changing some code, but not most or all of the code. Rewriting means all the code is brand new.

Based on what I've been told about the 16 bit IDE, it doesn't look like porting was ever on the table, at least not for the bulk of the IDE code. So that leaves rewriting, with all its attendant problems.

One of the best examples in C7 of the perils of rewriting is the window previewer. It wasn't there at first, and that caused a lot of problems because the window *designer* in C7 is actually .NET code. Among other things .NET handles containers differently than the Clarion RTL does. If you put a control on a tab in .NET, and the control is bigger than the tab, .NET clips the control to the container. The C7 runtime doesn't do that. For this and other reasons SoftVelocity eventually had to recreate the window previewer.

There are still features missing from C7 that annoy the daylights out of some Clarion developers. There are still, I'm sure, bugs in C7 that didn't exist in C6 and prior versions. Of course there are also many new features and benefits to C7, but it's what is missing or buggy that gets the users' attention.

Okay, that was the easy part

SoftVelocity has done a commendable job in persisting with C7, fixing bugs, adding and restoring features, and generally making C7 a more productive environment than C6 (yes, I know, some will disagree, but I think I'm on the majority side here).

Now comes the real work.

If rewriting Clarion for C7 was a huge task, to my mind the job of creating Clarion.NET, at least as SoftVelocity has described it, is bigger yet. And it's not just about the Clarion.NET AppGen.

As a Win32 tool, Clarion 7 enjoys a number of advantages, including:

- A single target platform - yes there are ASP and PHP templates, but essentially Clarion targets the desktop.
- A mature client/server application architecture
- Code generation benchmarks - just make sure the app generates the same code as it does in C6
- An existing 32 bit RTL and file driver system

In contrast, Clarion.NET needs a whole bunch of brand new code:

- There is no one standard architecture for .NET business apps - Clarion will be setting a standard (or standards) for Clarion developers
- The LINQ to File provider is brand new
- The Clarion RTL has been ported/rewritten, and will undoubtedly have some issues

As well, Clarion.NET is targeted at a minimum of four platforms:

- WinForms
- WebForms
- Compact Framework
- WPF (as of recent comments by Bob Z)

Presumably, at some point Clarion.NET will need to target Silverlight (which is a generally a subset of WPF), ASP.NET MVC, and perhaps Windows Phone 7 (which uses Silverlight). So that's as many as seven platforms.

All of this means integrating with various screen designers, after they become available in SharpDevelop. It also means creating application architectures which are multi-tier with logically separate user interface, business logic, and data processing code. I'm not aware that SoftVelocity has any special expertise in multi-tier architectures, so the company will have to get that somewhere.

Further, all of this work has to happen in the context of an IDE that *isn't* Visual Studio. And because it isn't Visual Studio you don't have Visual Studio's extensibility points and rich third party tool support. So you can't, say, enjoy the productivity benefits of tools like Resharper or CodeRush (not that either works with Clarion# anyway).

At the same time that SoftVelocity is trying to build a saleable .NET product it's also maintaining and enhancing the Win32 product. The company has also outsourced development, a process that while potentially saving money has arguably delayed the delivery of C7 and affected its quality.

What is the future of Clarion# and Clarion.NET?

I don't even know the current status of Clarion.NET with any certainty, so the future is absolute conjecture at this point. Take the following with a grain of salt.

I see Clarion.NET as a project with assets and liabilities.

Clarion.NET's liabilities include:

- The Clarion# language. The problem here is that there are next to no education resources out there that reference Clarion#. Compare that to the tens and hundreds of thousands of examples in VB.NET and C#. If you want to convert that code to Clarion#, well, you're going to have to understand the original anyway. So just get over it; pick one of the two main .NET languages and learn it. Or both of them. Most Clarion folk will probably find VB.NET to their taste; I'm comfortable with C syntax and I actually like case sensitivity, so I'm a C# fan. Now, having learned VB.NET and/or C#, what's your compelling reason to use Clarion#, which is just a thin veneer over C# anyway? Yes, it's a great way to bring in legacy code, but chances are building whole apps with it will be less satisfying than using one of the mainstream languages.
- The standalone Clarion IDE. It was a great idea at the time, but Visual Studio, especially 2010, is far, far superior. And with the extensibility points now available, a VS version of Clarion.NET code generation system seems eminently doable. If only it was like this in 2003.

Clarion.NET's assets include:

- The file driver system. I don't know if the LINQ to File provider is going to be the way you'll want to get at the file driver system, but direct access is definitely still an option. SoftVelocity could earn bonus points by providing C#/VB.NET sample code.
- The report engine. I have to admit I haven't tried the Clarion# report engine. If it works well, it would be a great way to preserve the massive investment many developers have in custom reports. Bonus as above.
- The runtime library. Queues in particular come to mind. Bonus as above.
- The code generation system (potentially). Code generation has always been Clarion's core strength, in my view. If SoftVelocity can come up with a well-designed, extensible layer on top of T4 (not necessarily a complete AppGen) I think that could find many uses, provided it lives inside Visual Studio.

Clearly there's potential value in Clarion.NET for Clarion developers who go on to .NET, even if they choose not to use the Clarion.NET IDE and Clarion# as their primary application development toolset.

Originally, the stated vision for Clarion.NET was a standalone IDE generating WinForms, WebForms and Compact Framework applications. Outside of the Clarion community, and to some degree inside of it, I don't think that vision has a ton of appeal these days. The .NET business software development world is moving on to WPF, Silverlight, and Windows Phone 7. Silverlight in particular is getting a lot of traction for its ability to build both intranet web and desktop user interfaces.

As I noted earlier, Bob Z has said Clarion.NET will have support for WPF applications. Both WPF and Silverlight are built on XAML, an XML-based declarative syntax for defining user interfaces. The latest SharpDevelop releases contain a WPF editor, and presumably this will be available to Clarion.NET at some point. How long it will take to integrate that editor into Clarion.NET, and how difficult it will be to integrate it with the AppGen, is something only SoftVelocity can answer (and maybe they don't know either).

So, does Clarion.NET have a future?

If I had a crystal ball, I'd be out fishing somewhere instead of writing articles about something as silly as programming. But okay, here's my take.

Yes, Clarion.NET has a future, but at present it's not a very bright one. There's no AppGen yet, and past performance suggests that even if one does appear shortly it will take a long time before its usable, particularly for newer technologies such as WPF.

Any widely acceptable application architecture for business applications will be different enough from the current client-server architecture that very little embed code will be directly usable in a .NET version of the same application. Even if the architecture were similar (not a good thing at all) most of the UI-related embed code wouldn't work anyway.

The Clarion Win32 templates are mature, with many third party products. A lot of apps rely on those third party templates. This kind of mature template environment could take years to redevelop for .NET.

It's hard to see Clarion# having much if any appeal outside the Clarion community, particularly as it's C# under the hood. And as C# gets new features, Clarion# will inevitably have to follow, which will eat up developer resources.

Clarion 7 will continue to consume resources that could be going into the .NET product.

Unless a great deal has been happening behind the scenes that we don't yet see, at the current pace I would expect Clarion to continue to lose ground as other .NET development tools improve.

Although not suitable for large app development, Microsoft has done some excellent work with Visual Studio LightSwitch. And there are other model-driven tools out there, such as GenWise Software Factory and Dawliasoft's latest version of Sculpture (with the unfortunate name of S-Expert). For that matter, hand coding business apps isn't as bad as it once was either. Heresy, I know.

For SoftVelocity's sake, and for the sake of their customers, I hope they've charted a path through all of these challenges and can deliver, in a timely manner, on their promise of highly productive .NET development.

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 Robert Wagner on September 7 2010 (comment link)

Hey Dave - you sure do know how to open a can of worms.

One worm you didn't mention re Clarion# is its apparent Russian pedigree. I surely hope that it works out OK for SV, but talk about an imponderable. The argument against rewriting the C5 IDE was multiple unsupported languages, lost or impenetrable source code, etc. Just think about the problem of going forward with something written in god knows what with Russian datanames and comments. And that's assuming that you can get your hands on the source code. All I can say is, good luck.

by Lee White on September 7 2010 (comment link)

Robert,

A lot of extremely good code comes from Russia and countries that resolved from the former Soviet Republics. Don't forget the RTL bug TopSpeed couldn't find that was reverse engineered by Alexey who sent them the fix. Honestly, in today's world, it makes no difference where the code comes from as long as it works.

I'm just saying.

---

by Dave Harms on September 7 2010 (comment link)

Robert,

Not enough worms for you, eh? :)

I don't think the code being written by Arcadia is necessarily any better or worse simply because it's written in Russia. But the fact that SoftVelocity has outsourced/offshored some significant part of Clarion.NET could certainly be a factor in the long delay.

Offshoring hasn't always worked out for Clarion in the past either. Yes, the London Development Center produced some very cool technology in the form of Clarion for Windows, and they did it in a relatively short timeframe. But these are also the people who left a legacy of unusable 16 bit Clarion code, and you can argue they share some of the blame for the current situation.

Dave

---

by Andy Stapleton on September 7 2010 (comment link)

thanks dave, I appreciate your insight...

---

by Bjarne Havnen on September 7 2010 (comment link)

I have done some extensive work in Clarion# mostly targeting web, and has come to the conclusion that unless you need Clarion FILE, it has nothing to that can't easily be replaced with C#. In my last project, I eventually had to pull out most of Clarion specific objects. I chose Clarion# because of FILE, QUEUE and EVALUATE, not to mention I had a lot of code I wanted to more or less copy-paste. Because of instability and lack of progress on reported bugs, I replaced FILE with SQL data objects, making classes that mirrored the file structure and gave me the necessary "ABC" methods. Evaluate was replaced with msscriptcontrol (with a little parsing).

You mention Queues, but using LINQ with enumerables does excactly the same.

Today, I have one project that has to use Clarion# because I use TopSpeed FILE. I believe that in the near future, I will port the entire project to VS, but leave a set of filemanagers for the TopSpeed files. Hopefully this will work just as fine as using Clarion# objects from C# projects inside the Clarion IDE.

In my opinion, the lack of progress for the Clarion.NET IDE and AppGen, suggest that SV would be better of writing a plugin for VS - or any .NET IDE, that enables the use of essensial Clarion stuff. Typically, an object that easily wrap a TopSpeed driver around a defined class structure (e.g file). Of course, I would love a code generator for my business objects, but I can do with making VS templates just to get the job done.

Also, the web is evolving at a pace that suggest the techology we used last year, will be obsolete next year, if not before.

---

by Arnor Baldvinsson on September 7 2010 (comment link)

Well said Dave!

I would like to point out to Robert Wagner, that the Clarion compiler has been maintained by a Russian, Alexey Solovjev, for the past 11 years or so;)

But I do agree that outsourcing the entire project, as I understand SoftVelocity did or at least most of it, was probably not a wise move. On their case (study page) Arcadia quote 54 months or 500 man-months as project time and that they started in June 2004. 54 months is four and a half years so they should have been done by the end of 2009 if I'm calculating correctly. However, SoftVelocity said in the end of 2006 that Clarion.NET would be ready for beta by the end of 2006 and for Gold in early 2007. I find that interesting information!

Best regards,

---

by Majodi Ploegmakers on September 8 2010 (comment link)

Well put Dave, that's roughly how I see it too.

Back in 2004 we knew we had to start working on replacement of our product suite that was developed in the 90's. So at the 2004 Devcon I had a (for me) important talk with Z about the future. I bought it (both the story and the product). So did others.

5 years later (2009) my worries grew to extreme. That’s because we started to examine the technology landscape as a start for our new product family and found that this landscape already changed dramatically in these past 5 years (including Visual Studio) but at the same time we were pretty much in the dark regarding Clarion.net progress.

In the past we always relied on Clarion to provide us the technology base (the plumbing) not to worry about, and building on top of that with our own domain specifics. So it’s important to know these base building blocks and to feel confident about it while waiting. But for that you have to communicate, talk. We are paying customers and, I think, have the right to some communication. Instead we get scarce blogs with not much meat, unanswered email and newsgroup postings are deleted! I still can not believe this.

So like you, we don’t have the details of where Clarion.net is going. We’re in the dark. Not only about the status, the progress or release planning but also about used technology and architecture (roadmap). SoftVelocity is becoming an unreliable partner for professional development if they don’t start working on all of this.

Currently we are forced to investigate other options like you’ve mentioned (Genwise, Sculpture and VS enhanced with tools from other vendors like Telerik).

I’m glad Clarionmag provides us an independent platform here. What I don’t understand is that (again) there is a big absentee here.

by Graham Dawson on September 8 2010 (comment link)

Hi Dave,

> You can see this for yourself in the softvelocity.public.clarionsharp newsgroup, where a number of requests for information have gone unanswered

Unfortunately that's just what you cannot see. Most of the recent posts asking for an update are almost instantly removed from the server. It's worth visiting www.clarion-software.com to see the full(ish) version of the newsgroup.

I think the low point was reached yesterday (7th September) when Geoff Bomford posted

> The IDE is available, there is just no appgen!
>
> 12 months ago I thought it might be ready by now, now I think it might be another 12 months. If you haven't bought it already, then I'd wait, and invest your time in learning VS2010.

This was 'edited' and appeared without the second paragraph ie just

> The IDE is available, there is just no appgen!

If it wasn't for Geoffs subsequent post no-one would know any different.

To me that is dispicable behaviour, deliberately altering the text of a post completely mis-representing the intent of the postee.

As for Clarin.NET I just wonder if the problem comes down to money, or rather the lack of it. Perhaps economics means Arcadia is no longer involved and it's all being done in-house, I don't expect we'll ever know.

I can see no reason for Clarion# or the Clarion.NET IDE to exist at all.

I **can** see a reason for a T4 based templated C# code generator plug-in for Visual Studio

Graham

by Dave Harms on September 8 2010 (comment link)

Thanks Graham. You're quite right - when threads are completely deleted there's no way for a new reader to see the lack of response, short of going to http://www.clarion-software.com/.

Editing a post without indicating that the post has been edited, and why, is inexcusable, regardless of any economic circumstances.

Dave

---

by Dave Harms on September 8 2010 (comment link)

Thanks Majodi. I only wish I had better things to say about the situation.

Dave

---

by Vince Sorensen on September 8 2010 (comment link)

It's apparent that Clarion.NET is DOA.

Once I figure out a nice way in VB.Net to replace Topspeed files without all the overhead of SQL, I will be moving my products over.

---

by William Tetley on September 9 2010 (comment link)

Dave, very honest and valid assessment. How about an article about Clarion 7? I was very high on Clarion 7 8-10 months ago (our whole programming staff was), but it seems like every new build introduces new problems and makes the product less desirable. We feel trapped, cannot go back to Clarion 6 and Clarion 7 does not seem to be the answer.

Tony

---

by William Tetley on September 9 2010 (comment link)

Also meant to mention that we have been using Clarion .NET for some components of our main program and works fine, but offers no real advantage over Visual Studio.

---

by Skip Williams on September 9 2010 (comment link)

This is one of the newsgroup messages that was quickly deleted...

Dave,

I feel the same way, Dave. Three years ago I spent a lot of money for a Clarion.net with Appgen and have been waiting ever since...actually since Z showed a working Clarion# compiler at ETC (2001?). It was running in their editor, producing IL code, and compiling into a working 'Hello World' Program.

Since that time it seems that there have been at least three major shifts in direction....moving to the Sharp Editor, shifting from compiling to IL to interpreting Clarion# code to C# code, then compiling the C# code to IL. Thirdly, moving appgen to T4. I'd be willing to bet that a lot of time had been spent on appgen before making the shift to T4. That is a lot of 'start overs'.

These major shifts have most likely caused the long delays to a public release, but getting next to no feedback on the progress is pretty disheartening to those of us who had to justify the expenditure to our management.

Skip

---

by Dave Harms on September 9 2010 (comment link)

Tony,

C7 is a bit of a mixed bag. I really do think that it's the answer for the majority of Clarion developers, based on conversations I've had and on the newsgroup postings. I don't think the reliability issues are on the whole any worse than they are for C6, and there are many productivity improvements. But not everyone is going to see it that way, and C7 isn't going to be the solution for all Clarion shops. And while we can at least all see that SV is working on C7, the fact that it isn't tighter after all this time is a concern.

I'll give another article some consideration - thanks.

Dave

---

by Dave Harms on September 9 2010 (comment link)

Skip,

Do you mean etc 2004? The 1.0 release of the .NET platform only happened in 2002.

I agree that the shifts in technology have probably caused some significant delays. But the scope of Clarion.NET is just way too big for a company the size of SoftVelocity. It may have looked like a manageable task at the beginning of the decade, but the .NET framework has grown into a huge platform since then, and SV is trying to compete on that scale. It just can't work.

While Clarion# (the language) is a nice idea, the more time passes the more I think it's a net (so to speak) liability, as is the standalone SD-based IDE. I have no doubt that if SV were starting this project today they'd do a VS plugin instead, and I would hope they'd abandon the idea of Clarion# as being too expensive with too little return.

Dave

---

by douglas johnson on September 9 2010 (comment link)

Dave,

-- But the scope of Clarion.NET is just way too big for a company the size of SoftVelocity.

I agree completely. Clarion.NET and even Clarion7 provide a classic case of a business not focusing on what it does best in order to provide the leverage necessary for a small company to compete.

Douglas

---

by Dave Harms on September 9 2010 (comment link)

Vince,

TPS is one of the very few business-scale ISAM file systems around today. Another little-known option is Microsoft's very own ESENT. There's also a managed code library. I haven't tried ESENT, but it is available on every Windows install starting from W2K on. It's also the foundation of the RavenDB document database, by Oren Eine (Ayende). If it's good enough for Ayende it must be pretty solid indeed.

Might be worth looking into.

---

by Jim Hedge on September 9 2010 (comment link)

Hey Dave,

How about a comment from Mr. Z?

jim

---

by Dave Harms on September 9 2010 (comment link)

Jim,

Bob Z's comments are always welcome here.

Dave

---

by Paul MacFarlane on September 9 2010 (comment link)

@Vince...

What about SQLite or embeded firebird as a replacement....

See: http://www.clarionmag.com/cmag/v9/v9n02sqlite.html and: http://www.sqlite.org/

---

by Dave Harms on September 9 2010 (comment link)

Paul,

Good possibilities for single user, just not multi-user.

Dave

---

by Jerry Greene on September 10 2010 (comment link)

Does it mean,if we don't talk about it, maybe it will just go away?

---

by Arnor Baldvinsson on September 12 2010 (comment link)

In response to questions on databases, there is a lightweight system from Sybase called Advantage. It's similar to SQL Anywhere, except it is completely free for deployment as far as I know. I haven't tried it as I have used TPS or XML for single user stuff. But it might be something to look at. http://www.sybase.com/products/databasemanagement/advantagedatabaseserver The local server is a single dll that you can distribute freely. They do have .net providers but I don't know if this is suitable. It was something I looked into 2-3 years ago as an option for simple, free, desktop SQL.

Arnor

---

by Vince Sorensen on September 13 2010 (comment link)

Seems like there are lots of options to look into ... thanks!

---

by Dave Harms on September 13 2010 (comment link)

Arnor,

I see that as of 9.10.0.9 Advantage has support for the Entity Framework. Definitely worth a look - I think anyone looking to do business software development for .NET should at least be considering either EF or NHibernate for data access. And although I've been using NHibernate, I think at this point I'd probably give the nod to EF for its Visual Studio tooling.

Dave

---

by Alex McCullie on September 15 2010 (comment link)

Thanks Dave for the reasoned speculation. That's all it can be outside of the SoftVelocity camp. I have given away all hope for Clarion.NET (and written off my initial dollar investment) as a competitive software tool in the .NET space. I'd hoped for an appgen capability which was Clarion's distinct advantage over other Win32 tools. Unlike most Clarion developers, I never held the language or the supporting ABC framework in particularly high esteem.

Today I work with VB.NET (and sometimes C#) every day in multi-layer web environments for intranets. Some projects are effectively hand-coded with supporting frameworks and others generated by the Ironspeed application generation. It depends on how closely the task fits a traditional CRUD model. As interest only I've played with the web aspects of Alpha from Alpha Software. There are lots of strong competitors!

To be honest I feel SoftVelocity's situation a bit sad, even though at times I wonder how they survived so long.

Alex

---

by Brian Reid on September 15 2010 (comment link)

Thanks Dave for a very good article. I think I have gone through all the grieving stages as it relates to Clarion.net. I have lost thousands of dollars in the multiple Clarion.net licenses we purchased from SV based on promises we received back in 2005.

The only question I have now is, how will SV treat the 100 or so customers that purchased Claron.net licenses when it becomes obvious to themselves that it was a lost venture. They will have to think about the $200,000 to $400,000 they took from their customers and left them with vapourware.

I'd be curious if anyone ever got their money back. Our shop has moved on to VS 2010. A bargain at a fraction of the SV price.

Brian

by Rhys Daniell on September 15 2010 (comment link)

Thanks Dave, I agree with every word.

You could sum it up this way: SV's meagre development resources are struggling to deliver C7, let alone keep up with an increasing flood of new technologies.

As I've said here before, my belief is that by the time Clarion.Net becomes a robust product, .Net itself will have been superseded.

I recall betting Z at the last Sydney Devcon 3? years ago that C7 would not go gold at the end of that year, as he was confidently predicting. Looking at the most recent bug list for C7.2 you would have to say that if the gold exists it is still significantly tarnished.

SV should stick to its core market of delivering Win32 databased desktop business apps. They could usefully provide improved IDE support for SQL back ends, and provide a way to develop Win32 thin clients (ClarionNet and HandyTools have showed the way) - while not as sexy as the alphabet soup referred to above, these would be of considerable value to the core of Clarion developers who are maintaining and extending existing applications.

You mention history. CPD was a success because Bruce Barrington and his team declined to be swept up by the latest technological fad and had a laser-like focus on producing robust business applications very quickly. When using CPD you were keenly aware that it was being used every day by members of the Clarion development team, meaning that the IDE was fast and intuitive.

The focus of the current team is very different.

by ANDREW BULLEN on September 15 2010 (comment link)

I agree with Rhys that SV should stick to its core market of delivering Win32 databased desktop business apps. This is still a huge market and will continue to be so for quite a while. We provide global applications and whether we use Citrix / RD or WEB - nothing beats a regular local Windows application for user performance.

A team of 10 Clarion developers can produce a better more maintanable system than 100 hundred .NET coders when you are talking about a databased desktop business app.

I am happy to write off my .NET investment with SV and pay them more money to make C7 as easy to use as C6 and for SV to continue to enhance and improve it.

SV have an awesome product in regular Clarion - keep improving it - relax and make some money.

---

by Abe Jimenez on September 16 2010 (comment link)

I would have a problem committing to Clarion.net even if they came out with a great version next week. I don't know how many people have purchased this product, but I know that it's a subset of the Clarion for Windows user base. The Clarion user base is already very small and not growing. I cannot see the whole world jumping on Clarion.net and making it a player in the .net world.

Without a much larger user base, I can't see how SV can come up with the resources to keep up with it's competitors even if they get through the hurdles of a first release.

If I were to recommend a Clarion.net web site to one of my clients I would have to commit to supporting them for the long term. That is just not a responsibility I want to take on when the underlying technology is as iffy as Clarion.net

---

by Dave Harms on September 16 2010 (comment link)

Alex,

> There are lots of strong competitors!

I agree, although I still wish there was an end-to-end code generation system for .NET that approached the functionality of Clarion in Win32. I haven't seen it yet. But there are a whole lot of tools out there, which is one of the reasons we now publish (shameless plug alert!) DevRoadmaps.com.

---

by Dave Harms on September 16 2010 (comment link)

Brian,

> I think I have gone through all the grieving stages as it relates to Clarion.net.

You're not alone.

> I'd be curious if anyone ever got their money back.

Honestly, I don't think there's any point in trying. Those of us who have been with Clarion for a decade or more (and that's most of us) have made far more with the product than we've lost by waiting for Clarion.NET (although I think I've probably taken as big a hit as anyone, and bigger than most). It's less trouble to take the write-off.

I think SV had the best intentions; it's just that from this vantage point it looks like it's all gone horribly wrong. I give them full points for still being here, and for continuing to work on C7, and I hope they are able to do that for years to come.

---

by Dave Harms on September 16 2010 (comment link)

Rhys,

> You mention history. CPD was a success because Bruce Barrington and his team declined to be swept up by the latest technological fad and had a laser-like focus on producing robust business applications very quickly.

BB was subject to grandiose visions too. I remember him announcing that the next version of Clarion would be the multi-platform version - Unix, Windows, maybe something else too. That never materialized, and the next version in fact was CDD, which almost killed the company.

Business is tricky, and reinventing a successful business is trickier. CPD was the right product at the right time; CDD was late to the DOS party. CW was a bit late to the Windows party, but still provided good value because it had some beautiful abstractions. But it never really grew beyond the Clarion community.

Out of the five main products so far, (CPD, CDD, CW, C7, Clarion.NET) CPD had, by my guesstimation, by far the largest market share with CW a distant second.

---

by Dave Harms on September 16 2010 (comment link)

Andrew,

> SV have an awesome product in regular Clarion - keep improving it - relax and make some money.

I think that would be a fine strategy. It'd be even nicer if they'd find some way of giving back to those who did buy the .NET product. I've said before that there are some pieces of Clarion.NET that could be useful to developers who otherwise do their .NET dev using Visual Studio and C# or VB.NET.

Dave

---

by Dave Harms on September 16 2010 (comment link)

Abe,

> Without a much larger user base, I can't see how SV can come up with the resources to keep up with it's competitors even if they get through the hurdles of a first release.

It is a little hard to imagine. I posted some things I thought they should do, but really, that's not my decision. I've retracted the comment.

---

by Michael Gorman on September 21 2010 (comment link)

I started buying Clarion in 1987. I kept buying it version by version for the next 11 year. But it was not until 1998 that I took out my wallet and spent well into the 6-

figures of my own money into the development of the Metabase system.

Why didn't I develop the Metabase system in earlier versions of Clarion? Simple. The cost of maintenance was going to exceed the cost of development.

I just converted the Metabase from 6.3 to 7.2 this past weekend. The conversion went very smothly. I have 16 apps with 3 dll-based apps. Across the 1200+ CLWs, there were less than 10 screen conversion errors. I fixed everthing in several hours.

Why didn't I convert from 6.3 to 7.0 two years ago. Simple. It wasn't ready and I had no compelling need for C7.0

Now C7.2 appears quite ready for "my" prime time and I'll be developing in C72 from now on.

So, why is this related to Clarion.Net? Simple. I see Clarion.Net in the same way as C7.0 two or more years ago. It's just not ready, and I have no compelling reason to attempt any Clarion.net development. So, I'll just wait till it's ready for "my" prime time.

Right now I'm helping an organization create proposals to the Fed for consulting and data management work. There's a group of dotNet developers right behind my cube. I listen to them as they do their work. At one point I asked whether it would take me more than "x" hours to accomplish something. The "x" corresponded to how long it would take me to develop a set of tables into a running app. They looked at me like I had 10 heads (not just two) and said that it would take many many times more than the hours I had suggested. I then showed them the Nurses application I created in about 300 hours for Maxine (daWoman) and they were totally blown away.

When I showed them the whole engineering of Clarion they couldn't believe what they were seeing. It was sort of like I was some alien from an IT civilization way in the future.

In the 23 years of Clarion purchases and evolution, I have never been ultimately disappointed. Have things taken longer? Sure. But that's the case with everything we do in IT. While I would like to believe that SV possesses all the characteristics of silver-bullet designers and programmers, I credit them with no more "seeing into the future" than I credit to myself.

So, I'm willing to wait. Better late right than early wrong. Besides, by the time Clarion.Net gets here will full code generation, these guys next to me will still be coding and building for themselves huge landfills of code that needs to be maintained by hand.

Regards, Mike Gorman

---

by ClarionMag user on September 21 2010 (comment link)

Mike, what you're talking about is something that gets discussed a lot by Clarion devs who do have a need to do .NET development right now.

There's no question in my mind that the kind of productivity we now enjoy with Win32 Clarion doesn't yet exist in .NET.

At the same time, productivity is far better in .NET these days than it was in Win32 C or C++ when Clarion for Windows first arrived. And there is a surprising amount of code generation in VS itself, just not the end to end kind we're used to.

But yeah, I really wish I had Clarion-style productivity in .NET. What I have isn't bad at all (and many of the tools are light years better than anything Clarion provides), but I want it all.

Dave

BACK TO TOP

Subscriptions extended by 30 days

By Dave Harms

Posted September 14 2010

This has been an extraordinarily busy year here at Clarion Magazine. At the beginning of the summer we launched our completely rebuilt ClarionMag web site. And today we've launched a second site, called DevRoadmaps, using the same technology behind this site.

(If you have any interest at all in .NET development, whether Clarion.NET or non-Clarion.NET, I encourage you to check out DevRoadmaps.)

All of that effort has taken a toll on this summer's publication schedule. We haven't produced as many articles as we'd intended, so we've extended all subscriptions that were active as of or after May 1 by 30 days. With both sites now live and no further launches in the immediate future we anticipate a return to our usual publication schedule. Thanks for your patience.

You can check the status of your subscription on the My ClarionMag page. And if you have any questions, please let me know.

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

Just launched: DevRoadmaps.com, your guide to .NET development!

By Dave Harms

Posted September 15 2010

I'm delighted to announce to Clarion Magazine's readers that the DevRoadmaps web site is now online! DevRoadmaps.com is a publication of CoveComm Inc, the same company that has been publishing Clarion Magazine since 1999.

What is DevRoadmaps?

DevRoadmaps grew, in part, out of a number of conversations I've had with Clarion developers who felt the need to move some or all of their applications to .NET. The common thread in these conversations was the difficulty developers face finding the right kind of information in the vast universe that is .NET.

Clarion development, for all its intricacies, is a pretty small world. For the most part, we build desktop build client/server apps using a language and a runtime built on top of the Windows API. But in .NET the options are truly mind boggling. There are multiple desktop, web, and mobile platforms. There are a great many different languages, tools, libraries, techniques, and practices. There is a bewildering array of resources, and the best ones are not always easy to find.

So how do you find your way in .NET development? You turn to DevRoadmaps.com.

Please visit this page on the DevRoadmaps site for a Clarion perspective on .NET development.

Other DevRoadmaps pages of interest:

- The home page
- Description and tutorial

DevRoadmaps is about .NET development in general, not Clarion.NET development in particular. Clarion Magazine is still the place to find Clarion.NET information (at such time as the .NET AppGen is available and usable).

Whether you're using Clarion.NET or non-Clarion tools, I believe you'll find

DevRoadmaps.com to be a vital source of information for your .NET business software development.

Beta special

Like ClarionMag, DevRoadmaps is a subscription-based publication. And there's some pretty good beta pricing on right now.

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 Paul MacFarlane on September 16 2010 (comment link)

Will there be any combo pricing between publications?

by Dave Harms on September 17 2010 (comment link)

Paul,

I will be announcing a cross-promotion today, in that after subscribing to DevRoadmaps at the current beta pricing you get a benefit you can apply to your ClarionMag subscription. It will be retroactive to those who have already subscribed to DevRoadmaps.

Dave

by Dave Harms on September 18 2010 (comment link)

Whoops, make that Monday

BACK TO TOP

CLARION MAGAZINE
READ. LEARN. SOLVE.

Home | Subscribe | E-Books | News | Blog | Store | My ClarionMag | My Lists | Contact

Customizing the Clarion 7 editor's syntax highlighting

By John Morter

Posted September 22 2010

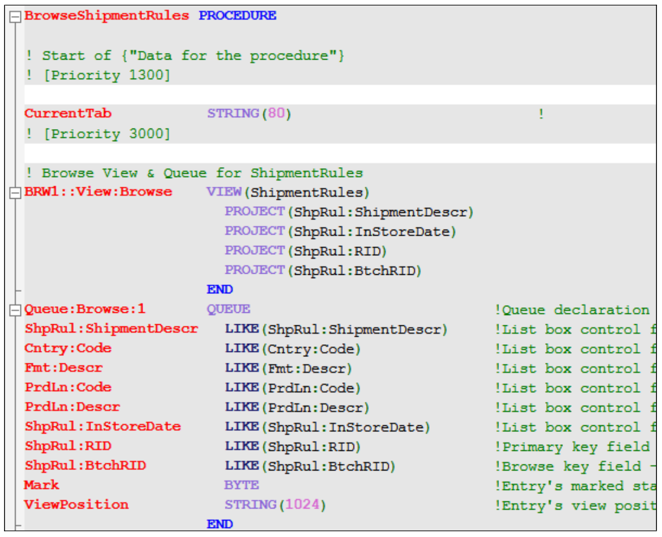
The first article I wrote for Clarion Magazine was titled "Customizing Clarion 5's Editor and Menus", way back in July 1999. The catalyst for my submission had been an earlier article by Bruce Wells explaining how to go about configuring the colour highlighting used by the Clarion 5 Editor.

Nothing much has changed in the past 11 years; I'm still taking inspiration from other people's examples.

This time it was the generous response by Lee White to a question I had posed on SoftVelocity's Clarion7 newsgroup that inspired me. I asked if anyone had found a better font to use with the C7 editor, in place of the standard Courier New (bor-ring!!).

The answers I received (see news.softvelocity.com/sv.clarion.clarion7, circa 14th July, 2010) included suggestions for quite a few font alternatives – plus something that really caught my attention; a screenshot resulting from Lee's editor configuration, which immediately got me wondering how he'd achieved it.

Comparing figures 1 & 2, you'll see why I was keen to emulate Lee's implementation.



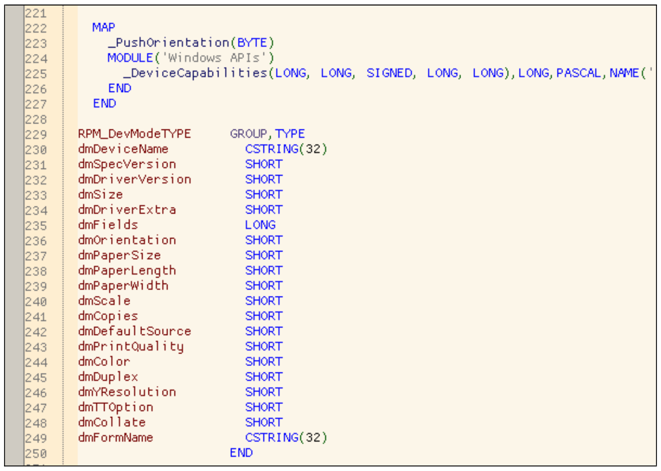
```
BrowseShipmentRules PROCEDURE

! Start of {"Data for the procedure"}
! [Priority 1300]

CurrentTab           STRING(80)                          !
! [Priority 3000]

! Browse View & Queue for ShipmentRules
BRW1::View:Browse    VIEW(ShipmentRules)
                       PROJECT(ShpRul:ShipmentDescr)
                       PROJECT(ShpRul:InStoreDate)
                       PROJECT(ShpRul:RID)
                       PROJECT(ShpRul:BtchRID)
                     END
Queue:Browse:1       QUEUE                            !Queue declaration
ShpRul:ShipmentDescr   LIKE(ShpRul:ShipmentDescr)     !List box control f
Cntry:Code             LIKE(Cntry:Code)               !List box control f
Fmt:Descr              LIKE(Fmt:Descr)                !List box control f
PrdLn:Code             LIKE(PrdLn:Code)               !List box control f
PrdLn:Descr            LIKE(PrdLn:Descr)              !List box control f
ShpRul:InStoreDate     LIKE(ShpRul:InStoreDate)       !List box control f
ShpRul:RID             LIKE(ShpRul:RID)               !Primary key field
ShpRul:BtchRID         LIKE(ShpRul:BtchRID)           !Browse key field -
Mark                   BYTE                           !Entry's marked sta
ViewPosition           STRING(1024)                   !Entry's view posit
                     END
```

Figure 1. The standard C7 Editor configuration (with Courier New font)



```
  MAP
    _PushOrientation(BYTE)
    MODULE('Windows APIs')
      _DeviceCapabilities(LONG, LONG, SIGNED, LONG, LONG),LONG,PASCAL,NAME('
    END
  END

RPM_DevModeTYPE     GROUP,TYPE
dmDeviceName          CSTRING(32)
dmSpecVersion         SHORT
dmDriverVersion       SHORT
dmSize                SHORT
dmDriverExtra         SHORT
dmFields              LONG
dmOrientation         SHORT
dmPaperSize           SHORT
dmPaperLength         SHORT
dmPaperWidth          SHORT
dmScale               SHORT
dmCopies              SHORT
dmDefaultSource       SHORT
dmPrintQuality        SHORT
dmColor               SHORT
dmDuplex              SHORT
dmYResolution         SHORT
dmTTOption            SHORT
dmCollate             SHORT
dmFormName            CSTRING(32)
                    END
```

Figure 2. As included with Lee's response to my NG question

It didn't take too much "poking around" to work it out, generally - but I was still intrigued by Lee's specific configuration. And I recognized, even in his short/small example, an eye for good design and colour coordination. At this stage I did consider asking my wife for some help, but I decided it would take a lot less explaining to simply ask Lee for a copy of his configuration file, which he was happy to send me, hence my reference above to his generosity (thanks once again, Lee).

My 1999 Clarion Magazine article explained how to go about customising the key mapping used by the C5/6 text editor; a fantastic capability that has been lost with the "advance" to C7!

This little article is limited to coverage of the ability to configure the C7 editor's colour highlighting ('cos that's about all we're now able to do … *sob*!), and explains how I followed Lee's example – in case you'd like to do so too.

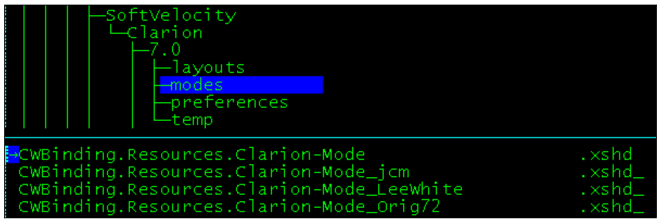
Before you start though, and for your own protection, please check out the contents of your "mode" folder, which is where any pre-existing Editor colour-configurations will be stored.

If you have a Windows XP environment then you'll find this folder via `C:\Documents and Settings\<UserName>\Application Data\SoftVelocity\Clarion\7.0\modes`.

In a Windows 7 environment, you'll find it via `C:\Users\<UserName>\AppData ~\SoftVelocity\Clarion\7.0\modes`.

If you do find a file therein named `CWBinding.Resources.Clarion-Mode.xshd` then you have already created an override to the standard colour highlighting, and you should take a backup of this file before proceeding.

Figure 3 is an example of the "modes" folder from my PC, showing various backups I've taken by the simple expedient of adding some filename suffixes.

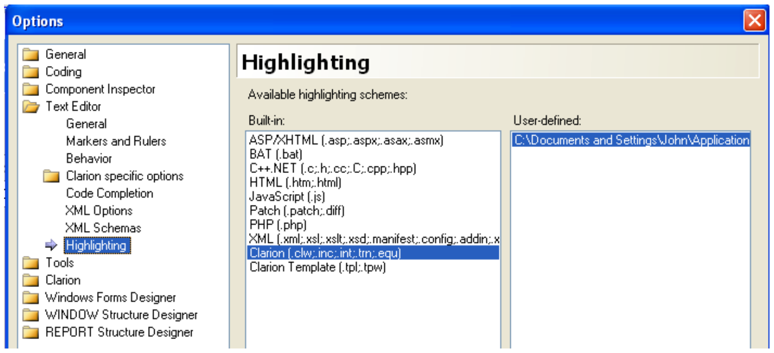


```
│  │  │  ├─SoftVelocity
│  │  │  │ └─Clarion
│  │  │  │   ├─7.0
│  │  │  │   │ ├─layouts
│  │  │  │   │ ├─modes
│  │  │  │   │ ├─preferences
│  │  │  │   │ └─temp

CWBinding.Resources.Clarion-Mode                    .xshd
CWBinding.Resources.Clarion-Mode_jcm                .xshd_
CWBinding.Resources.Clarion-Mode_LeeWhite           .xshd_
CWBinding.Resources.Clarion-Mode_Orig72             .xshd_
```

Figure 3. The "modes" folder, in which editor colour-configurations are stored

As intimated above, there's a simple process to override the colour highlighting used by the C7 editor; You've probably already noticed it as one of the settings available via "Tools \ Options" (off the main menu), where it's cleverly labeled "Highlighting".


Options

- General
- Coding
- Component Inspector
- Text Editor
  - General
  - Markers and Rulers
  - Behavior
  - Clarion specific options
  - Code Completion
  - XML Options
  - XML Schemas
  - Highlighting
- Tools
- Clarion
- Windows Forms Designer
- WINDOW Structure Designer
- REPORT Structure Designer

Highlighting

Available highlighting schemes:

Built-in:

```
ASP/XHTML (.asp;.aspx;.asax;.asmx)
BAT (.bat)
C++.NET (.c;.h;.cc;.C;.cpp;.hpp)
HTML (.htm;.html)
JavaScript (.js)
Patch (.patch;.diff)
PHP (.php)
XML (.xml;.xsl;.xslt;.xsd;.manifest;.config;.addin;.x
Clarion (.clw;.inc;.int;.trn;.equ)
Clarion Template (.tpl;.tpw)
```

User-defined:

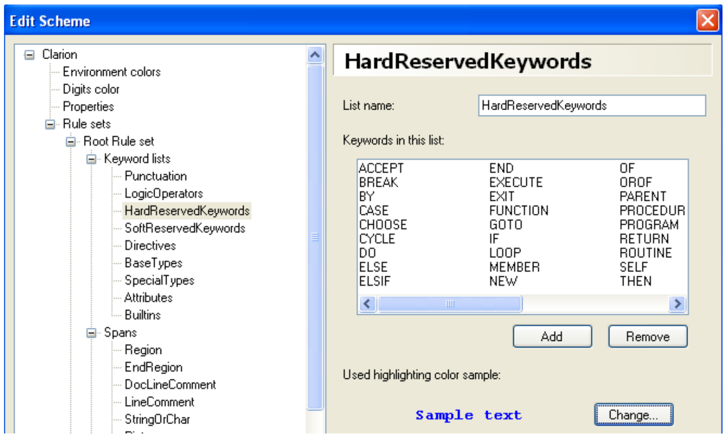
```
C:\Documents and Settings\John\Application
```

Figure 4. C7 IDE Text Editor Highlighting

The process of creating an override is straightforward enough; you simply select one of the built-in highlighting schemes (in this case, Clarion or Clarion Template) then click the "Copy to user-defined" button … and a copy of the standard colour highlighting scheme is created as a User-Defined scheme..

After doing this, you'll find a new file in the `SoftVelocity\Clarion\7.0\modes` folder (named with a `–mode.xshd` suffix). You might like to make a backup copy of this file, as I did, as a reference point back to the standard scheme configuration.

Now, click the "Modify" button and you'll be presented with an "Edit Scheme" window, via which you can change colour settings and make choices about which keywords are to be associated with particular groups of colour settings.


Edit Scheme

- Clarion
  - Environment colors
  - Digits color
  - Properties
  - Rule sets
    - Root Rule set
      - Keyword lists
        - Punctuation
        - LogicOperators
        - HardReservedKeywords
        - SoftReservedKeywords
        - Directives
        - BaseTypes
        - SpecialTypes
        - Attributes
        - Builtins
      - Spans
        - Region
        - EndRegion
        - DocLineComment
        - LineComment
        - StringOrChar

HardReservedKeywords

List name: HardReservedKeywords

Keywords in this list:

```
ACCEPT      END         OF
BREAK       EXECUTE     OROF
BY          EXIT        PARENT
CASE        FUNCTION    PROCEDUR
CHOOSE      GOTO        PROGRAM
CYCLE       IF          RETURN
DO          LOOP        ROUTINE
ELSE        MEMBER      SELF
ELSIF       NEW         THEN
```

Used highlighting color sample: Sample text

Figure 5. Example of ability to change colour highlighting

At this point, I wouldn't blame you if you began to wonder what our esteemed Editor was doing when he allowed this lame article to be published, but there is a little gem that I'm working towards (albeit, slowly) … so I'll lift the pace, and it may be worth your while to hang-in.

I was intrigued by a couple of things: Firstly, I wondered about the difference between `HardReservedKeywords` and `SoftReservedKeywords`… and secondly, I soon lost track of the changes I'd made to different groupings of Clarion Commands within each keyword category. I realized I needed a "bigger picture" perspective. And the *best* way to achieve this is to open the `~mode.xshd` file directly, outside the IDE, using your favourite text-editor.

Note: If, when you do so, you encounter a very long string, starting with "Syntax Definition", then you'll need to first click the OK-button from within the "Edit Scheme" window … to force the IDE to flush-out the scheme into XML format.

You should now find it much simpler to see what's where – and easier to swap keywords around between groupings, just so long as you're careful not to damage any of the XML tags as you go.

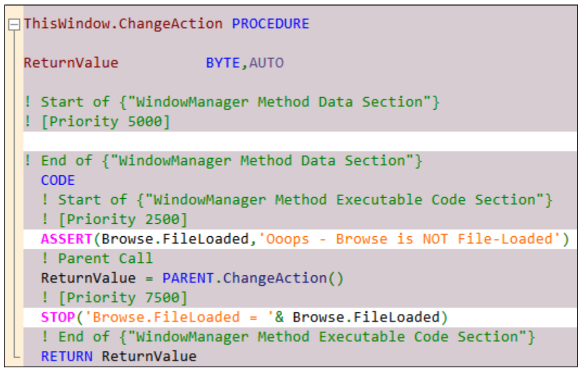
Here, though (and *finally*!) you may be saying), is the bit that I thought might be useful to my fellow Clarioneers.

I found that; i) the naming of the keyword groupings is unimportant (i.e. you can change the name of, say, `HardReservedKeywords` to anything you wish, without any repercussions that I could detect), and ii) additional keyword groupings can be added – also without any adverse results.

Why would I want to do either of these things?

Because choosing my own keyword group names allows me to be more descriptive about why I chose to group keywords in a particular way and in terminology that's meaningful to me. And, creating new groups altogether allows me to have more, and more flexible, colour assignments.

For example, I actually do find the debugger to be useful, but I also find the good-old `STOP()` to be simple and effective. However, I want to be sure I can spot these `STOP`s in my code (mainly so I don't mistakenly leave 'em in there!). To do so, I created a new keyword group that I've named "Highlights" and associated with a bright fuschia/magenta colour. Keywords I've added to this group include `STOP` and `ASSERT` … with the effect shown in Figure 6.



```
ThisWindow.ChangeAction PROCEDURE

ReturnValue          BYTE,AUTO

! Start of {"WindowManager Method Data Section"}
! [Priority 5000]

! End of {"WindowManager Method Data Section"}
  CODE
  ! Start of {"WindowManager Method Executable Code Section"}
  ! [Priority 2500]
  ASSERT(Browse.FileLoaded,'Ooops - Browse is NOT File-Loaded')
  ! Parent Call
  ReturnValue = PARENT.ChangeAction()
  ! [Priority 7500]
  STOP('Browse.FileLoaded = '& Browse.FileLoaded)
  ! End of {"WindowManager Method Executable Code Section"}
  RETURN ReturnValue
```

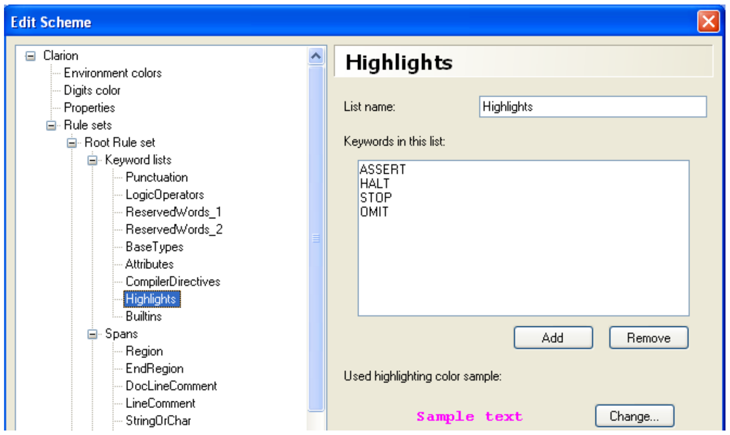
Figure 6. Example of my customised "Highlights" group

Here's the configuration in my `CWBinding.Resources.Clarion-Mode.xshd` file that results in this effect:

```
<KeyWords name="CompilerDirectives"  bold="true" italic="false" color="#574684
       <Key  word="BEGIN" />
       <Key  word="COMPILE" />
       <Key  word="INCLUDE" />
       <Key  word="ITEMIZE" />
       <Key  word="ONCE" />
       <Key  word="SECTION" />
</KeyWords>
<KeyWords name="Highlights"  bold="true" italic="false" color="Fuchsia">
       <Key  word="ASSERT" />
       <Key  word="HALT" />
       <Key  word="STOP" />
       <Key  word="OMIT" />
</KeyWords>
<KeyWords name="Builtins"  bold="false" italic="false" color="#8000FF">
       <Key  word="FALSE" />
       <Key  word="NULL" />
       <Key  word="TRUE" />
</KeyWords>
```

Note: If you happen to list the same keyword twice, within different keyword groups, then the last-listed instance takes precedence.

The Tools | Options “Edit Scheme” window now happily includes my self-created “Highlights” group as one of the keyword lists … after all, it’s simply parsing the XML in the `CWBinding.Resources.Clarion-Mode.xshd` file.



Highlights

List name: Highlights

Keywords in this list:

ASSERT
HALT
STOP
OMIT

Used highlighting color sample: Sample text

Figure 7. My self-created “Highlights” group is now one of the Rule sets.

That’s it! … I hope this contributes to clear and easy-to-read code in your projects.

To get you started, you will find my current `CWBinding.Resources.Clarion-Mode.xshd` file as a download from this article … just be sure not to unintentionally clobber yours with it.

Download the source

John Morter is a member of the Victorian Clarion Users Group (Melbourne, Australia). John is Asia Pacific IT Manager for a brand-name multi-national and he's supposed to leave all the fun technical stuff for others to do. So, his Clarion work is developed under the nom-de-keyboard Flat Chat Solutions, where "flat chat" is an Australian expression meaning doing something at top speed / high velocity.

Article comments

by S Jayashankar on September 23 2010 (comment link)

Hi!

Is there any way of changing the highlighting of the brackets i.e. currently it is Dark Blue on a bluish-gray background with a box around it. This is visually jarring for me and obscures the cursor. I would prefer it to be just Red on white like Notepad++ or other editors.

Regards

---

by Bob Roos on September 23 2010 (comment link)

I tried your color scheme. One thing I noticed is that the "current line marker" that used to be yellow is now a deep red. The rest of the coloring was OK, but the generated code was darker background than I like.

In the "Environment Colors" section I changed the "Generated Code" color to a light green. That seemed to cause the "Selection" color to change and that seemed to change a lot of other colors as well. A comparison of the file from this article and the one on my disk shows that in addition to the changes I made, the "Custom Name" of "Generated Code" was dropped.

It looks like after "Generated Code" was dropped all the colors moved up a notch. That is the category "Generated Code" on the left was gone and everything moved up a line, but the colors stayed the same. Is there perhaps an XML error in the .xshd file you have with this article?

---

by Bob Roos on September 23 2010 (comment link)

Also, what is the "current line marker" called? I couldn't find it to change its color.

Thanks, Bob Roos

BACK TO TOP

CLARION MAGAZINE
READ. LEARN. SOLVE.

Compiling Clarion Source To Java Bytecode

By Andrew Barnham

Posted September 23 2010

Clarion deserves to be deployed on a modern managed code platform such as Java or .NET, because the design goals of Clarion and Java / .NET overlap and complement one another. The promise of Rapid Application Development is furthered by being delivered on a technology platform designed to provide reliable, safe, and managed computing.

Of course, Clarion# is already available for .NET. But Clarion# is not the same language as Clarion. The languages share superficial syntactic similarity yet there are significant differences in the fundamentals of how a rich client application is expressed. The `WINDOW` structure and `ACCEPT` loops are gone, and database access differs substantially at a code level. Clarion# may be all well and good and an optimal choice for a brand new project, but what about those amongst us that have existing Clarion code, and those amongst us who place value in the years of investment and knowledge that is bottled in that source code?

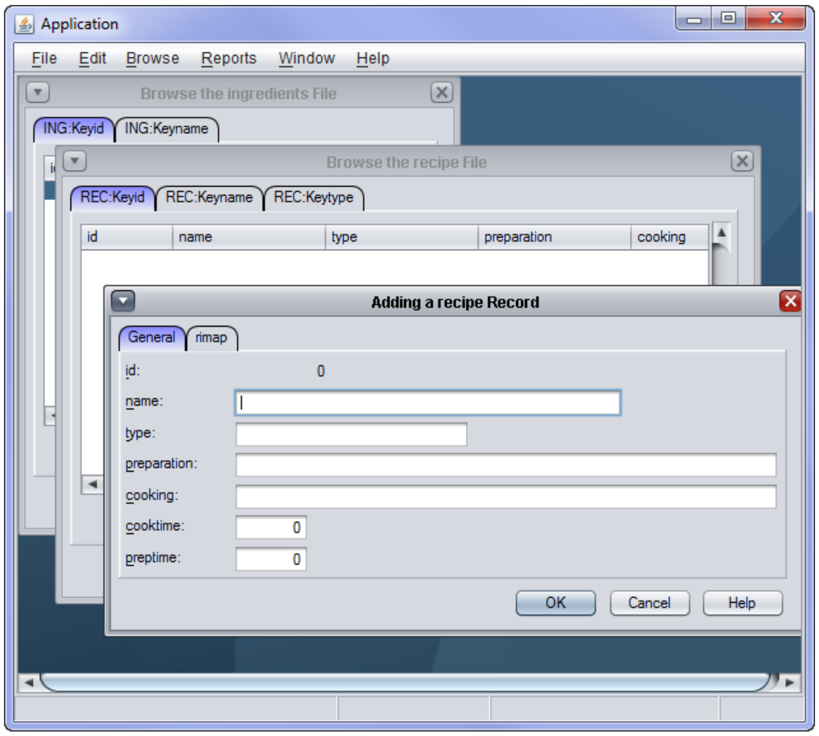
Also consider end users who have grown accustomed to our applications behaving in certain ways, with certain precise keystroke combinations used for repetitive data entry. Going to a new system, such as Clarion#, means painstaking work to ensure controls behave the same way or and that our programmatic interceptions such as `?button{prop:disable}=true` are faithfully replicated. Or alternatively it means pushing further technology disruption onto end users by forcing them to relearn a new look and feel. “The old system let me do X, but I cannot do it with the new system. You call this an upgrade?”

If there is no forward path that permits us to gracefully modernise our existing software assets, and if the only option is to rewrite, then why limit our thinking to just Clarion#? If we are forced to go through the pain and disruption of a significant rewrite then why not cast a wider net and also consider something completely new – like VB.NET or some other competing RAD platform?

Rewriting applications in Clarion# or some other .NET language is one option; another is porting the Clarion language so that the application code can be used unchanged.

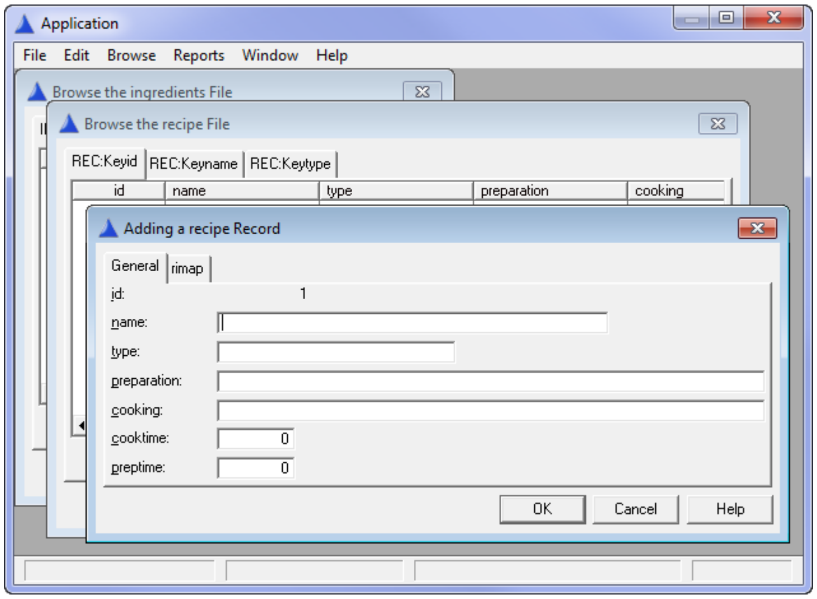
In this article I'll briefly describe Clarion2Java, an open source project that demonstrates that a graceful path forward into a modern era of software engineering is indeed available to Clarion programmers, by compiling Clarion source code into Java byte code.

Figure 1 shows one of two sample APPs, the CookBook application, compiled for Java. Figure 2 shows the same application compiled as a normal Clarion application.



id: 0
name:
type:
preparation:
cooking:
cooktime: 0
preptime: 0

Figure 1. The CookBook Java app



id: 1
name:
type:
preparation:
cooking:
cooktime: 0
preptime: 0

Figure 2. The CookBook Clarion app

What modern engineering techniques bring to RAD

The case for Clarion in a managed environment, be it Java or .NET, is based on a number of key points

- Memory management. Memory management goes hand in hand with the business programming philosophy. Clarion programmers only want to worry about the business logic, not about internals of system memory architecture and having to code explicitly for allocation and de-allocation of memory. Why should a Clarion programmer have to worry about the unpredictable consequences of illegally referencing unallocated memory, or having to walk through a queue of ANY objects in order to de-allocate the ANY objects so that the program does not 'leak' memory? Clarion should take care of these concerns on behalf of the programmer (as it once did). The benefit of memory management has been discussed in a number of IT articles (such as Joel Spolsky's How Microsoft lost the API War) and it is regularly recognized as the single most beneficial productivity driver for programmer efforts over most other trends in the common professional practice in the past 20 years.
- A stable run-time environment that provides useful feedback when there are fatal software logic errors. A managed enviroment can report to a programmer the exact line of code that failed the nature of the failure and a full trace of 'stack' activity which can integrate with a powerful step debugger. Compare this with GPF failures which yield nothing useful.
- An environment that is popular amongst the broader IT community for which hundreds of thousands of libraries and packages exist to solve common problems.

The key unifying theme at work here is increased productivity or, as it is also known, Rapid Application Development. Productivity is not the only gain with running on a managed environment either.

Why not .NET?

Why did I choose Java over .NET? No reason in particular, other than I have been working with Java for 10 years so this allowed me to develop Clarion2Java faster than I could develop Clarion2DotNet. Java also has a few minor advantages: it is more mature, is closer to an open-source model, cross-platform and is a more popular language. Yet if expertise was not a consideration it would be difficult to single out either .NET or Java as target platform. Both would serve well.

Clarion2Java is an open source project that demonstrates that all of the above is theoretically possible and, as a working reference implementation, very much achievable. It permits Clarion source programs, without modification, to be compiled and deployed into the Java environment. Clarion programs thus compiled take advantage of all the features and richness of the Java environment delivering the key bullet points above and yet still permitting the Clarion programmer to continue to program in Clarion and continue to use the source code developed to date.

What exactly is Clarion2Java?

Clarion2Java is a compiler that takes Clarion source code and compiles/converts it into Java source code which can then be compiled into Java byte code.

Consider a very trivial example Clarion program (which ships with Clarion2Java)

```
    program

    map.
    code

    message('Hello World','Example')
```

This is compiled by Clarion2Java into Java source code resembling this:

```
package clarion;

import org.jclarion.clarion.Clarion;
import org.jclarion.clarion.crash.Crash;
import org.jclarion.clarion.runtime.CRun;
import org.jclarion.clarion.runtime.CWin;

public class Main

{

    public static void init()
    {

    }


    static
    {
        init();
    }



    public static void destroy()
    {

    }


    public static void main(String[] args)
    {
        try {
            init();
            begin(args);
            CRun.shutdown();
        } catch (Throwable t) {
            Crash c = Crash.getInstance();
            c.log(t);
            c.crash();
        } finally {
            destroy();
        }
    }


    public static void begin(String[] args)
    {
        CRun.init(args);
        Cwin.message(Clarion.newString(
            "Hello World"),Clarion.newString("Example"));
    }

}
```

The above may seem to be such a lot of verbose code for generating a simple hello world, and in some respects it is. Yet remember that Java is a general purpose language, not a business language so it will generally be more verbose when it is applied to solve purely business concerns. The point of Clarion2Java is that you do not need to worry too much about the output, it is generated automatically and you need not consult it. The above is provided purely to give a flavour of what Clarion2Java does under the hood.

This code is in turn compiled into Java executable which can be deployed into a Java virtual machine.

Knowledge of Java code is not necessary for a Clarion programmer. You continue to program in Clarion.

The process

The process for Clarion programmers can be illustrated as follows. Figure 3 shows the Clarion .exe work flow.

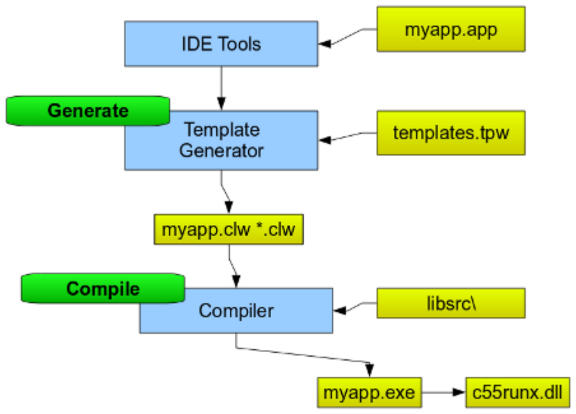


Figure 3. The Clarion workflow

The Clarion2Java work flow starts off the same way, but changes at the compiling stage:

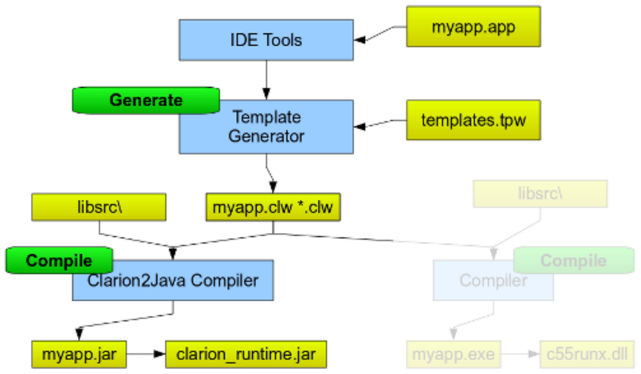


Figure 4. The Clarion2Java workflow

The Clarion2Java compilation process is done outside of the Clarion IDE. After code generation you run a Java build tool which compiles the generated source into Java, and packs the resulting Java files into a JAR file (which is in ZIP file format). For a complete discussion of the compilation process please see Section 3 in UserGuide.pdf, which is contained in the Clarion2Java download.

Clarion2Java will compile most any Clarion code, including hand written Clarion source code and template-generated code for any template chains including legacy, ABC and other third party templating systems.

As noted above, Clarion2Java also ships with two simple app generated examples, one based on Clarion 5.5 and one based on Clarion 6.0.

How complete is Clarion2Java?

The Clarion2Java implementation handles a very large sub set of the Clarion language specification, but being a relatively young open source project is it not 100% complete. As I write this there is no file driver support for TPS or Clarion(DAT) databases. Database access in Clarion2Java is provided via JDBC access. Clarion2Java is heavily tuned to work with PostgreSQL databases only and the implementation takes advantage of PostgreSQL-specific features for purposes of optimising database performance. Generic support for JDBC is planned.

Other file drivers are supported, such as BASIC, ASCII and DOS. In addition to this, the file drivers system for flat file based access is managed by a extensible system that permits introduction of extensions so that file drivers can seamlessly handle things such as auto-inspecting and decompressing, .ZIP files, auto-converting .XLS files into .CSV, and even internet resources such as files denoted by http:// or ftp:// names. Memory based files, via mem:// are also supported.

The implementation is generally sufficient enough to permit deployment of very large scale applications into the several hundreds of template generated procedures and modules. I successfully manage a 400 procedure Clarion application as a Clarion2Java project.

Broad concerns such as windows, reports, queues, groups, files, system functions are all supported in Clarion2Java and exactly as they are coded and behave in Clarion.

The system also supports Clarion style reports, with all of the pagination, header, group, tallying and orphan control systems that exist in the original Clarion language. Clarion2Java internally encodes pages as .PDFs, and offers a report preview capability.

All other windowing structures and styles, menus, browses, and forms all compile and work under Clarion2Java, including MDI style applications.

Clarion2Java supports a multi-threaded model and implements the Clarion 6 threading style. `THREAD` properties behave exactly as they do in native Clarion .EXEs.

Clarion2Java also provides additional extensions and capabilities which do not exist in Clarion such as the ability to import Java third party libraries. You can also manipulate window design and layout while the application is running and take the result of that and re-import it back into the compilation process.

At time of writing, there are a number of small limitations and incomplete implementations in Clarion2Java. For example:

- Clarion 5.5's cooperative threading model is not formally simulated. So Clarion 5.5 apps need to be careful about multiple threads access shared memory
- The window `REGION` control is not yet implemented.
- `POPUP()` is not implemented
- Some functions are not thread safe. e.g. `EVALUATE`, `BIND` and `ERROR`.

A full list of incomplete implementations can be found on Clarion2Java's open source page on SourceForge. All of the items above are easily and readily resolved, but Clarion2Java being a volunteer effort requires volunteers or financial support of volunteers to see the edges rounded on the project.

Constraints and limitations

There are some things Clarion2Java will probably never support because they cannot be implemented with pure Java code. These include:

- Integration with third party DLLs
- Support for OCX/ActiveX controls

These limitations generally should not represent a significant barrier to entry, as the Java community is a rich community which hundreds of thousands of third party libraries (most of them free to access and free to use) will allow you to substitute existing DLL integrations with Java equivalents.

Clarion2Java implements `OVER` and `PEEK` and `POKE`, but Java is a strict memory managed language and support for these controls is implemented by simulating how x86 memory in a Clarion application is laid out. The system works quite well and provides the flexibility that functions such as `OVER` provide and the safety of a memory managed language, yet it comes at a performance cost. Usually the cost is negligible and difficult to detect, but some code segments relying on `OVER` will notice this, particularly more complex concerns such as overlaying large `GROUP` structures where 'memory contents' changes rapidly. Such code will benefit from rewriting to use solutions that avoid excessive use of `OVER` attributes. The same principal applies for doing reference associations (`&=`) for structures which are not polymorphically related.

More benefits

Being open-source provides a number of benefits, including the ability to modify the source code yourself and add your own features or at least debug and understand how the code is behaving. One of the key success drivers behind Java was that the core source code libraries were available for programmers to inspect. The original license for Java did not permit modification or distribution of these source libraries (upcoming Java 7 will be largely open-source) , but as a programmer, having access to this code is invaluable component of the job. Sometimes, no matter what language you are using, things do not work the way you expect. For instance, a fatal system error could be emitted from deep within the run-time libraries your application depends on. Having access to source code permits you, when all else fails, to trace into the library code and try and understand what is going wrong. But with a black box you have no recourse but to request technical support from the library provider. Your business is dependent upon the support of a single gatekeeper. At least with open-source, even if you may not understand the system yourself you can always find someone who does and hire their services.

Java also provides a number of benefits beyond what has already being mentioned. Of notable value is the ability to run Java on non-windows platforms, such as Mac or Linux. In fact, Clarion2Java was largely developed on Linux systems with testing performed on Windows systems. Platform interoperability has been a design goal of Clarion2Java from its inception.

Additionally, the range of programming languages that can run on Java has expanded considerably over the years. You can mix and match Java source code with Clarion source code with code from a number of other open source projects dedicated to bringing choice and diversity to the Java managed environment. Languages such as JRuby, Jython, Scala, Groovy, Lua, Ada, Cobol, PHP, Pascal among many others are all available in the Java managed environment. People who develop on Java are spoilt for choice, and now can add Clarion as a another string in their bow via the Clarion2Java project.

Is performance an issue?

Since its initial inception in the mid 90s, Java has had a reputation for being slow. This may have been the case with version 1, but with recent versions it's a completely different story. For a number of reasons and technological and hardware advances, Java keeps parity with its compile time cousins and in many circumstances can even outperform them.

As an example of Java performance, jake2 is a Java implementation of the Quake 2 3D graphics engine. Performance in the latest version is neck-and-neck with the original C code.

My experience with deploying Clarion2Java applications is that most users report a perception of significant performance improvement from going from the Clarion .exe native system to Clarion2Java. The graphics do not feel quite as 'snappy' as the original .exe – as Java does its own window drawing – yet everything else, database access, browsing, updating records, printing reports are all reported to perform substantially faster.

One cost however is that Java is more resource intensive. It uses more memory and it uses more CPU. The end user experience is faster, but Java demands more resources in order to deliver this experience. On reasonably modern desktop equipment, anything from the past five years, Java runs smoothly and effortlessly without taxing total available system resources.

Getting Clarion2Java

Clarion2Java can be downloaded form source forge here:

http://www.sourceforge.net/projects/clarion2java/

The download includes pre-compiled examples, user guides and instructions to help you get started.

Article comments

by Steven Parker on September 23 2010 (comment link)

Excellent, Andrew, and very interesting.

I look forward to reading more about the product and the progress you make.

by Dag Roger Sundmyhr on September 24 2010 (comment link)

WOW! Thanx alot!!

Then the question is:

Why didnt SV do it like this with Clarion# instead of creating a new langauge? As you say, why should I learn Clarion# ? 1. C# is much more than Clarion wil ever be 2. Visual Studio 2010 beats the crap out of #Develop (Clarion IDE), and its free.

Microsoft is in fact the creator of both Visual Studio, C# and the .NET Framework. How can SV Compete with that? They simpley cant. SOme years ago, my dream was that SV would give us the .NET compiler, letting us compile .NET assemblies instead of Win32 binaries. They could still use some kind of IMPORT(assembly) to let us use 3rt parties, but instead they started to create something completly new. Will it ever be finnished? Who knows, but I dont have the time so I had to learn C# instead. Will I ever touch Clarion#? Probably not. The one and only reason to even consider using Clarion# is because of TPS files.

Anyway, good work. I hope someone, sometime, will do the same thing with .NET

by Dave Harms on September 24 2010 (comment link)

Dag,

> Why didnt SV do it like this with Clarion# instead of creating a new langauge?

You'll have to ask SV that. But I'm not sure that this approach is as applicable in .NET as it is in Java.

By creating Clarion2Java, Andrew gets a significant benefit: multi-platform support. That offsets the fact that the application architecture hasn't changed - it's still a decades-old client-server design (one that has, to be sure, served Clarion developers very well).

In .NET a straight port of the Clarion language would lack many of the modern features of .NET development including language improvements, WPF, Silverlight, mainstream ORMs, etc. And it would still be a Windows-only approach (although possibly something could have been done with Mono). So the core benefit would be moving your apps to .NET, but they'd look pretty much like they do now (they'd almost certainly use WinForms, not WPF), and they'd be the same client server architecture.

Perhaps that's a sufficient benefit - I really don't know. But if it walks like Win32, and talks like Win32, why not just stick with Win32 and use interop? (Some RTL tools to make interop with .NET code easier would be a fine addition to C8.)

Much of the appeal of .NET, at least to me, is that it enables new and better ways of building apps. Unfortunately many of those ways are at odds with the traditional Clarion development approach. I can't really fault SV for trying to embrace that brave new world, even if the results so far have been less than we all hoped for.

I think it comes down to this question: Should Clarion.NET be an easy migration path for existing apps where the app code and architecture is essentially unchanged, or should it be a melding of Clarion's strengths (modeling and code generation) with the many benefits of front-line .NET development? Clarion2Java is an example of the former (for Java of course, not for .NET), while Clarion.NET is, as near as I can tell, an attempt at the latter.

---

by Andrew Barnham on September 24 2010 (comment link)

Hi all. Thanks for interest in this project.

Until SV engage us in dialog about their decision making processes, can only speculate on architectural assumptions and goals SV have used to guide them to date. I think such a conversation would be invaluably constructive for all parties involved. Dave's article on Future of .NET is quite good and nuanced: one gets the impression that SV are stuck between a rock and a hard place, or at least trapped inside a Warren Zevon song.

I find myself disagreeing with Dave and agreeing with Dag here. My motto for architecture works is "evolution not revolution". If I was architect at SV I would of made all possible effort to preserve the language and provide a multi phase approach that walks my existing, loyal, customer base through the technology transition with as little disruption as possible (which was exactly what I did with my customers and clarion2java). The new language approach would of been last resort only: only activated after some very serious consideration. Approach I would of taken (and did take with my end users, substitute "language" for "user experience" same thing applies):

1. Keep language as is as much as possible and compile to new platform. As I have demonstrated this is achievable without difficulty.
2. Work on extending the language to expand into new areas of opportunity. Support existing constructs but put in place support for new constructs and define migration path/patterns to allow programmers to incrementally embrace new design patterns.

Creating a new language is a fraught endeavour and needs to be done as a last resort only: especially if it means discarding a language like clarion which is well designed and thought out (except its object model). I am doubtful that in order to unlock new capabilities (all of them only new human UI channels), that it was strictly necessary to abandon the existing language. The existance of projects like nettalk shows that you don't need to throw the baby out with the bath water. Not providing a migration path just alienates your customer base and it is difficult in my mind at least to treat this strategy in sympathetic terms.

I think an interesting exercise would be to reflect back on the CPD to CW transition days. UI change from DOS/text to Windows was quite radical and pretty much required a rewrite for us at least and it took us a while to completely execute it. For about 2 year we ran DOS + Windows versions of the app side by side and slowly migrated more and more features across. But the language was largely preserved, and my recollection at least was that it was reasonably well done. (I was quite young at the time though, a junior programmer so interested to see what others percieved this period in Clarions history).

But thing here is that from rich client point of view there is no radical departure from Win32 to .NET. Rich clients still look and feel the same on both sides of the Win32/.NET fence. What is happening in industry is opening up new UI channels such as the Web. It is not a fundamental change in computer/human interaction : it is an extension of new features that not everybody necessarily needs or wants. New paradyms may encourage restructuring of code to encourage reuse of code such as business logic for example, but it should not be mandatory.

Platform interoperability was not the core reason why I created clarion2java. The java decision was made purely because of time/money constraints. I needed to generate a solution in a hurry: my customers were waiting and starting to show signs of impatience. I know both Java and clarion intimately and I was confident that I could finish the build in about 6 man months of solid effort. With .NET would of taken alot longer because needed to factor in learning of .NET. All my customers run Windows (about 40+ now are running clarion2java version of the app and they are all happy with the new app).

Reasons for c2j:

- Clarion 5.5 app was crashing all the time; deep in the RTL and appeared to be related to graphics performance. I wanted a proper debugger and I wanted to be able to provide a more stable runtime system for customers. I wanted a stable RTL and I did not get the impression from SV as of late that they were a good strategic partner in providing me this. So I decided that I wanted to go it alone. (I tend to do this alot in my career, and other professionals criticise me for it from time to time for being to quick to reject "consensus" solutions in favour of building something fit-for-purpose. I accept their criticism is valid but it is the way I prefer to work and it is a strategy that has been successful for me; which my critics acknowledge)

- No confidence that 6 or 7 would deliver the goods and fix bugs. Still Win32 afterall; could not get a changelog out of SV to figure out if my problems had been addressed.
- Outgrown .DAT files. Wanted/needed an ACID database. Never wanted to switch to TPS because loss of things like CFIL and CSCN.
- switched to postgres, but unhappy with 5.5's SQL integration for a number of reasons. SQL statements generated were inefficient and setting up ODBC clients was too complicated. Support and maintainence of the application got excessively complicated
- Wanted to unlock new features. And better network integration overall. i.e. why can't I use "HTTP://" as a file name with a DOS file driver?. Simple, little, things that unlock opportunities but do not require a change in the language.
- Tried to engage SV sales team to see what they could do. Completely underwhelmed by the attention they provided me in terms of helping me map out a path forward for the technology I manage. They were happy to reply to my initial emails with sales sound bites but as soon as I started asking more involved questions, normal and reasonable stuff for technology due diligence, they stopped responding to my emails.

---

by ChrisP> on October 11 2010 (comment link)

Is it possible to have the export work on an Android phone? Are there any developers that have approached this platform in Java?

---

by Andrew Barnham on October 11 2010 (comment link)

Export to Android has not been tested and it is unlikely that it will work.

Key issue is that Clarion2java is coupled to the Swing graphics library. This library does not exist in Android, Android has an alternative library designed for the phone which has completely different API.

Other than the Swing issue, I would expect the code to work, but it has not been tested.

Making Clarion2java 'android ready' and able to do something interesting in Android would not be alot of work. Compiler and runtime are largely in place. Only need to extend runtime in order to come up with a system that maps to how Android thinks about human UI, to compliment how desktop UI runtime currently implemented.

BACK TO TOP

CLARION MAGAZINE
READ. LEARN. SOLVE.

Home Subscribe E-Books News Blog Store My ClarionMag My Lists Contact

Still Staying Connected On The Road

By Ned Reiter

Posted September 30 2010

This is an update of an article first published in 2002.

If you're like a lot of Clarion software developers, you spend at least some time on the road, and, you no doubt have your favorite way of staying connected to the online world. If you stay in hotels, or with friends or family, a cable, DSL, WiFi or even phone connection is never far away. But what if you travel by recreational vehicle (RV)? In 1997, my wife and I bought a 38' motor home, and we've been on the road full time ever since. I've learned a lot in that time about getting online; in this article, I'll explore the current connection options available to RVers.

I won't go into detail about the various ways of getting online eight years ago but will address the options that are available today to the RVer. If anyone is interested in the historical methods, the original article is still available.

For RVers who stay in one location for a month or more, many campgrounds have the camp sites wired for telephone service – you only need to contact the local telephone company to order a phone line. This can get expensive if you move frequently, as installation charges run about $50 every time you order. You may even be able to get DSL or cable internet service in a campground, but it may require a one year contract, not too useful for someone that likes to travel. Another option that is becoming more common is WiFi internet access. Some campgrounds have excellent, professional installations with repeaters to cover the entire facility, but often the WiFi is a wireless router in the office and you can connect only if you're in a very close camp site or right outside the office.

With the advent of the cellular digital data services, connectivity has improved significantly. With a USB connected cellular modem, 3G data speeds are available in all but the most remote areas. Many use their internet connection while traveling down the road. Not the driver of course, but the passengers can have full internet access at 60 mph. Plugging a USB cellular modem into a wireless router like one of the Cradlepoint models allows multiple computers and printers to connect wirelessly to the internet and each other. Some of the current models of smart phones include the ability to tether, or connect to a computer, and be used as a cellular modem. For example, a Motorola Droid from Verizon Wireless has a data plan with unlimited data and through the use of a third party application can be tethered to a computer via USB and used as a cellular modem. How long this will be allowed by Verizon Wireless remains to be seen. Another example is the next version of the Android operating system; most Android based phones will have a WiFi access point feature, allowing several computers and printers to connect to the phone and share the internet connection. This feature will have a monthly cost and bandwidth limit.

The cellular modems and tethered phones are excellent solutions for many RVers but if internet access is needed everywhere, satellite is still the only solution.

In 2002, Motosat announced they were developing a mobile internet access product, the DataStorm. This is a true two way satellite internet system, using HughesNet from Hughes Network Systems. Motosat's contribution is an autolocating mount for the satellite dish. The system was in beta test for quite a few months, but was released to dealers earlier that year. It does require professional installation and isn't cheap. The hardware is available from under $5000. Installation will cost about $1000.

The Motosat/HugheNet system is quite a change from cell phones and land lines. The download speeds typically reach or exceed 1Mbps, and upload speeds reach 200kbps and up. As with any broadband connection, it's always on. Operation is easier than hooking up a modem. After parking your RV and leveling it, press the **Search** button on the controller. About five minutes later, you're connected to the internet. The system uses NAT (Network Address Translation) so your PC has a non-routable address, making you invisible to anyone outside of your local network. It's just like being behind a firewall and in fact the modem has a firewall and all ports are closed to the outside world. You can connect the satellite modem to a router, so you can share the connection with other computers on your wired LAN or via WiFi.

In addition to the hardware and installation costs, the service is $79/month or more, depending on the level of service needed and if you want a static IP address. For most RVers, the minimal service at $79/month is the right choice. While HughesNet has good service in all of the lower 48 states, coverage in much of Canada is less complete. In general, the closer you are to the border (where most of the Canadian population lives) the better the coverage. As you go further north you may need a larger dish to get a reliable signal. You can view HughesNet satellite coverage maps here. However, there are other satellite internet providers in Canada.

I have used the Motosat/HughesNet system for nearly ten years and I can't imagine ever going back to a dialup connection. We still have two different cell phones and service plans, but they're mostly used for voice calls now. I do use a Droid with tethering for a backup to the satellite connection, and campground WiFi if it's fast and reliable. With the satellite system, I don't even need to be in a campground. I can run the system from my batteries and inverter while parked in the remotest location.

As you can see, the choices are varied in both usability and cost. If all you need is email and some web browsing, then a cellular connection or campground WiFi is more than adequate. For internet access in most places and while in motion, a cellular modem is an excellent solution. But if you need a full time internet connection and the ability to transfer large quantities of data, then the Motosat/HughesNet satellite system is the only solution available today that will work wherever you may be parked. While I've used HughesNet throughout this article because that's the service we have, there are other providers of satellite internet that use the Motosat hardware. For example, if you need reliable Voice Over IP, then iDirect is a better choice than HughesNet. In general these other providers are more expensive, but they do offer more features.

With the rapid changes in technology, there will undoubtedly be even better, and faster, internet connections available to the traveler in the future. Some of the technologies to watch are 4G cellular, WiMax, LTE and some new satellite based services using a constellation of satellites in low orbits. All these promise to bring better and cheaper universal internet access.

Ned Reiter has been programming since 1961 and a Clarion user since CPD 2.0. After spending the first 53 years of his life in the Milwaukee, WI, area, he and his new bride, Lorna, moved to Palm Springs, CA, in 1995. Deciding that life in CA wasn't to their liking, the sold nearly everything and bought a 38' motor home in 1997 and went on the road full time. They are still enjoying life on the road while Ned works from the motor home, writing mostly custom Windows applications in Clarion for Windows.

Article comments

by Wolfgang Orth on October 9 2010 (comment link)

All I need now is a RV... Thx for sharing your experience - Wolfgang

BACK TO TOP