

Clarion Magazine

Clarion News

- » [amazingGUI 1.2.0.3](#)
- » [Molebox Intro Price](#)
- » [Thin@ 1.24](#)
- » [Super Import-Export 7.00](#)
- » [Super QBE 7.00](#)
- » [Super Import-Export 7.00](#)
- » [Noyantis nBit HTML Editor Wrapper Template](#)
- » [DMC Blog Update](#)
- » [CHT Build 14A.00](#)
- » [Tagkeys updated with new release of vuFileTools](#)
- » [PDF-XChange Viewer Update Recommended](#)
- » [SCA Micro Templates 2.02](#)

[[More news](#)]

- » [Clarion.NET FAQ](#)
- » [Clarion# Language Comparison](#)
- » [The Future of Clarion, Part 2](#)
- » [The Future of Clarion, Part 3](#)

[[More Clarion & .NET](#)]

[[More Clarion 101](#)]

Latest Free Content

- » [Source Code Library 2009.12.31 \(1999-2009\) Available](#)

Save up to **50% off ebooks.**
Subscription has its rewards.



Latest Subscriber Content

The Future of Clarion, Part 1

Many questions remain unanswered on the .NET front. When will there be a usable AppGen? The new template language is based on Microsoft's T4, but what will the implementation really look like? How will Clarion.NET support Windows Presentation Foundation and Silverlight, the current state of the art in .NET user interface development? How does the Clarion# language compare to C# and VB.NET? Dave Harms looks at these questions, and reveals a surprising fact about Clarion# that changes everything.

Posted Tuesday, January 12, 2010

Source Code Library 2009.12.31 (1999-2009) Available

The Clarion Magazine Source Code Library 1999-2009 full release is now available. Source code subscribers can download the installer from the [My ClarionMag](#) page. If you're on Vista or Windows 7 please run Lindersoft's [Clarion detection patch](#) first.

Posted Wednesday, January 13, 2010

Sneak Preview: The New ClarionMag Site

We're hard at work on the new ClarionMag site. Check out some screen shots and read about the new features.

Posted Thursday, January 14, 2010

The Future of Clarion, Part 2

What's the hottest technology in the Microsoft application development stack? Arguably it's Silverlight, and Silverlight is a subset of Windows Presentation Foundation, or WPF. Dave Harms looks at how WPF fundamentally changes Windows development.

Posted Monday, January 18, 2010

Keeping Separate Clarion Development Environments

If you're a contract software developer, you probably run into situations where you develop software for different clients using different versions of templates or tools, and perhaps different versions of Clarion. Ben Dell shows how to keep one complete copy of your environment for each project.

Posted Monday, January 18, 2010

The Future of Clarion, Part 3

Last fall SoftVelocity announced its planned architecture for .NET applications. How does SV's strategy dovetail with the state of the art in .NET application development? What kinds of choices face Clarion developers? What *is* the future of

» » [The Future of Clarion, Part 3](#)

[\[More free articles\]](#)

Clarion Sites

Clarion Blogs

Clarion in .NET?

Posted Tuesday, January 19, 2010

ClarionMag Private Beta Update, And Some SQL

The new ClarionMag site has gone into private beta. David Harms discusses the process of setting up the site on a beta server, and looks at some complicated SQL code to fix a seemingly simple bug.

Posted Thursday, January 21, 2010

Clarion 7.1 Build 6695 Release Notes

Clarion 7.1.6695 has been released, with some important bug fixes and a few new features. Here's a quick list; we'll have a review next month after the ClarionMag office reopens.

Posted Thursday, January 21, 2010

[\[Last 10 articles\]](#) [\[Last 25 articles\]](#) [\[All content\]](#)

Source Code

The ClarionMag Source Code Library

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

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

[More info](#) • [Subscribe now](#)

Printed Books & E-Books

E-Books

E-books are another great way to get the information you want from Clarion Magazine. Your time is valuable; with our [e-books](#), you spend less time hunting down the information you need. We're constantly collecting the best Clarion Magazine articles by top developers into themed PDFs, so you'll always have a ready reference for your favorite Clarion development topics.

Printed Books

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



» » [Clarion Tips & Techniques Volume 5 - ISBN 978-0-9784034-1-6](#)

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

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

From The Publisher

About Clarion Magazine

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

Subscriptions

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

Satisfaction Guaranteed

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

Dave Harms

ISSN

Clarion Magazine's ISSN

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

About ISSN

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

Copyright © 1999-2009 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited.

Clarion Magazine

Clarion News

[Search the news archive](#)

amazingGUI 1.2.0.3

amazingGUI 1.2.0.3 has been released. Changes include: Fix for multi-DLL applications; A Clarion 7.1 installer. amazingGUI is available on Clarion Shop for \$119.

Posted Thursday, January 21, 2010

Molebox Intro Price

A 15.00% relative discount on Molebox is available until January 31, 2010. MoleBox is a software virtualization and protection tool that allows delivering your application as a portable stand-alone secure EXE file which runs instantly with zero installation. MoleBox packs all application files into a single efficient executable file that works without extracting packed files to the hard drive and creating temporary files. MoleBox also applies a number of protection techniques to packed files, including anti-crack protection for EXE and DLLs, resource protection, protection from modification for data files, and many more.

Posted Thursday, January 21, 2010

Thin@ 1.24

Thin@ 1.24 is now available. changes include: Clarion 7.1 is now fully supported (for Clarion7.0 thin@ files check .zip version of the installer; Command line scripts have been added to the setup dialog in the NetMonitor service (The default script allows starting / terminating thin@ service (NetServer and NetListen) if they become unexpectedly unavailable); Client side progress dialog function implemented; Prop:maxwidth and prop:maxheight attributes supported (allowing scrollable images); PressKey() and Press() function is now fully supported (6.3 on); Use ENTER keys instead of TAB template feature is now fully supported; Supported prop:join and prop:spread sheet's parameters; Spin controls up and down arrows now properly generate event:newselection event; Added functions Risnet:GetClientPath() AND RisNet:SetClientPath(STRING); Skip print preview report option is now working properly with Legacy templates

Posted Thursday, January 21, 2010

Super Import-Export 7.00

Super Import-Export 7.00 is available for download. It's a major upgrade, featuring: Job support to remember field assignments, automatic imports and exports, etc; Local variables can be included in list of exportable columns; User-specified formulas supported in exports; Extra #EMBEDs around list queue population; Improved handling of various unusual field and string delimiters; Add unlikely option of omitting fields when they are blank; FormatExportRecord CASE structure is broken into smaller chunks, to speed up compiles for very large export lists (and to enable exceedingly large export lists); When auto assigning fields using header record, matcher tries field names without prefix, uses UPPER, etc; Added

ImportBasic_PreviousRecord function, as well as support for ?PrevRecord button; Increased maximum limits on record size, field size and field count.

Posted Thursday, January 21, 2010

Super QBE 7.00

Super QBE 7.00 is available for download. It's primarily a compatibility upgrade, making it work with Clarion versions 6.x through 7.1.

Posted Thursday, January 21, 2010

Super Import-Export 7.00

Super Import-Export 7.00 is available for download. This is a major upgrade, featuring: Job support to remember field assignments, automatic imports and exports, etc; Local variables can be included in list of exportable columns; User-specified formulas supported in exports; Extra #EMBEDs around list queue population; Improved handling of various unusual field and string delimiters; Add unlikely option of omitting fields when they are blank; FormatExportRecord CASE structure is broken into smaller chunks, to speed up compiles for very large export lists (and to enable exceedingly large export lists); When auto assigning fields using header record, matcher tries field names without prefix, uses UPPER, etc; Added ImportBasic_PreviousRecord function, as well as support for ?PrevRecord button; Increased maximum limits on record size, field size and field count.

Posted Thursday, January 21, 2010

Noyantis nBit HTML Editor Wrapper Template

Noyantis Software has released a wrapper template for the nBit HTML Editor ActiveX control. The wrapper template enabled you to add a fully functional HTML Editor / Viewer into you application in just a couple of minutes. The capabilities of the nBit ActiveX control include: A simple word-processor type interface that presents a very slight learning curve for your end-users; Image uploading and manipulation; Hyperlinks; Cascading Style Sheets; Formatted HTML Code editing; HTML Table editing; An integrated FTP component for uploading images inserted into HTML documents; Customizable Toolbars; HTML DOM Tree Node Selector - Select any node of the current HTML DOM node and either edit the tag directly, set the CSS class or, in the case of TABLE elements, edit the layout properties of tables, rows and cells on a custom properties form; Switchable WYSIWIG view and raw HTML view; Spell Checker (depending on the existence of MS Word). The wrapper template includes both a template interface and Class methods to access all of the above as well as a Viewer Only mode where the Editor turns into a Web Browser complete with a Navigational Controls template option. Version 1.00 beta is now available for purchase from Noyantis, Clarionshop or Motleysoft. The template can be purchased on its own for \$90 or complete with the nBit HTML Editor ActiveX control for \$130 (providing a saving of \$29.95 if purchased separately). The template is c55, c6, c7, ABC and Legacy compatible. Example apps are included (Legacy app is still to be added to installer but will be available very shortly).

Posted Thursday, January 21, 2010

DMC Blog Update

A new DMC blog entry is available showing version 2 progress and details. A feature survey is also available.

Posted Thursday, January 21, 2010

CHT Build 14A.00

CHT Build 14A.00 is now available. Some 60+ templates were either added or revised in 2009. Considering that most of these templates also use at least one underlying CHT class, that means that there were also lots of new and revised classes added during the course of 2009.

Posted Thursday, January 21, 2010

Tagkeys updated with new release of vuFileTools

For users of Tagkeys there is a new import file for vuFileTools 3.5. This Tagset has been completely redone and includes all vuFileTools functions (more than 150) and all of the variables needed for each function. Tagkeys not only inserts the function, it also shows the calling parameters needed and automatically highlights them for you (for easy replacement). Tagkeys is a Keyboard / Clipboard text insertion and management tool. It allows you to type tags in your documents (Word, Notepad, Clarion editor, etc.) and have those tags replaced by any text you want. A free version of Tagkeys is available.

Posted Thursday, January 21, 2010

PDF-XChange Viewer Update Recommended

Important Security Update issued 30 December 2009: After consultation with Secunia Research (<http://www.secunia.com>) Tracker Software Products Ltd has issued an update to all versions of its Free/Licensed PDF-XChange Viewer product family including both end user and developer product ranges. On December 29th 2009, Secunia notified Tracker after testing, of the potential for malicious code to be executed to compromise users systems via PDF files, using a very specific means - though no such exploitation is known to have occurred to date. Within 24 hours an update was issued to all products blocking any such opportunity and all clients and developers are urged to update any versions of our PDF-XChange Viewer product to ensure that their systems are safe guarded. All PDF-XChange Viewer and Viewer SDK versions prior to release 2.044 should be updated to a later release.

Posted Thursday, January 21, 2010

SCA Micro Templates 2.02

SCA Micro Templates 2.02 is now available; Changes include: A new template for sending browse data to Excel using Clipboard; Invisible Calculator and Cleartype workaround now support Edit-in-place; New local templates to disable code generation for one procedure; Templates tested for compatibility with Clarion 7.1 and installer updated.

Existing customers can log in and download the update. The templates are US\$39 and this price includes a one year subscription to receive new releases. All source code is included (no black boxes).

Posted Thursday, January 21, 2010

Huenulefu Name Change

As of Wednesday, December 23 2009, Huenulefu Development SRL officially no longer exists. From now on, the company name is LARO Group SRL.

Posted Thursday, December 31, 2009

Clarion.NET Build Released

SoftVelocity has released an update for Clarion#. This release features the new LINQToFileProvider along with a few examples.

Posted Thursday, December 31, 2009

RPM for C7.1

A new RPM template and library install is available for C7.1. This is an interim release that does not support local(Lib) compiles. A full install is in development. In the meantime, if you want to do internal tests you can compile as standalone(DLL).

Posted Thursday, December 31, 2009

vuFileTools 3.5 Beta

vuFileTools 3.5 Beta is available for download. This release contains a number of bug fixes and new functions, including: vuBIOSSerialNumber; vuCRC32; vuAlreadyRunning; vuCPUUsate; vuClientWidth; vuClientHeight; vuReplaceCharsInFile; vuScreenDensity; viPrintTextFile; vuIsOS64. Be sure and read the help files as some functions have changed and can affect your current programs.

Posted Thursday, December 31, 2009

C7.x Third Party Deployment Script

Lindersoft has published a new "Clarion Accessory Deployment Demo.sb7" example script to demonstrate how to install third party products to Clarion 7.0 or Clarion 7.1. This project requires SetupBuilder 7.1 Build 2819 pre-release - do not use it with any previous SetupBuilder 6.x or 7.x version.

Posted Thursday, December 31, 2009

C7.1 Installation Test Tool

Lindersoft has released a first version of a test tool to check the C7.0 and C7.1 installation environment status. This freeware tool can be used to verify the installed Clarion 7.x version(s). If your auto-detection of Clarion 7.x fails, you can use this little tool to find out what the user has (or has not) installed. This is a code-signed application developed with SetupBuilder 7.1. Feel free to distribute it to your customers.

Posted Thursday, December 31, 2009

CPCS 7.10 Installers

Corrected Installers for CPCS version 7.10 and all Addon Products for the current Clarion 7 build are now available from the CPCS website. This build is free of charge to all users who previously purchased CPCS v7.0). Each of the addon product upgrades is free of charge to all existing registered users of any prior version of the same addon product. You will need to have CPCS v7.10 installed as well. These new builds fix the problem in previous v7.10 builds where the DLL and LIB files were named incorrectly causing errors when compiling and linking.

Posted Thursday, December 31, 2009

KSpng 7.1 Compatible

A Clarion 7.1 version of KSpng class is now available. KSpng class allows Clarion programmers to display PNG and TIFF format images using native Clarion IMAGE control on both windows and reports. Files can be loaded both as disk files and via URL from the internet. It also allows you to convert files between different image formats. BMP, GIF, JPEG, TIFF and PNG formats are supported. Everything is done with one call. Clarion 5.5, 6.3 and 7.1 both Legacy and ABC are supported.

During the beta period the product cost is \$69.95.

Posted Thursday, December 31, 2009

SetupBuilder 7.1 Pre-Release

The SetupBuilder 7.1 pre-release is now available. To get access, please send your SetupBuilder 7 serial number to sales at lindersoft.com.

Posted Thursday, December 31, 2009

CalendarPro Wrapper Template 2.02

Version 2.02 of the CalendarPro Wrapper template is available. Changes include: The detection of Event moving, changing and resizing has been enhanced; A new Event Exception facility has been added to the Recurring Events feature; Codejock 13.2.2 compatibility added; Allow Recurring Events to be Moved option added; Allow Recurring Events to be Resized option added; Icon sizes on events can now be specified; Event Selection option added to Right Click facility; Max No. of Selections option added to DatePicker; Detection of Event / No Event selection enhanced; New Calendar method: AllowRecurEventMove; New Calendar method: AllowRecurEventResize; New Calendar method: DeselectAllEvents; New Calendar method: GetEventOrigEndDate; New Calendar method: GetEventOrigEndTime; New Calendar method: GetEventOrigStartDate; New Calendar method: GetEventOrigStartTime; New Calendar method: GetEventLocation; New Calendar method: GetEventMasterEventID; New Calendar method: SelectEvent; New Calendar method: SetCaptionButtonText; New Calendar method: SetEnabled; New Calendar method: SetEventException; New DatePicker method: GetSelectionBlockCount; New DatePicker method: SetMaxSelection; GetEventAllDay method returned an incorrect value when the event was an AllDay event; SetDateRange method parameters changed from STRING type parameters to DATE type; Incorrect time values could be calculated depending on the user's Locale settings. The new version can be downloaded from the Members area using the original download and registration details contained in your sales email.

Posted Thursday, December 24, 2009

Locus Templates C7.1 Compatible

Locus templates have been updated to C7.1 compatibility (ABC templates only). If anyone wants the legacy templates updated, please contact Ben.

Posted Thursday, December 24, 2009

CPCS 7.10

CPCS 7.10 for the current Clarion 7 build is now available from the CPCS website. This build is free of charge to all users who previously purchased CPCS v7.0. Also, all CPCS Addon products for the current build of C7.1 are now available for download. Each of the addon product upgrades is free of charge to all existing registered users of any prior version of the same addon product. You will need to have CPCS v7.10 installed as well.

Posted Thursday, December 24, 2009

Fomin Report Builder 3.12

Changes in Fomin Report Builder 3.12 include: New single installation package for all Clarion versions (5.5, 6.1, 6.2, 6.3, 7.0 and 7.1); Support for run-time entry fields added (example reports changed to show this feature); Refined RTF input and

output support; Numerous changes and fixes. The internal implementation has changed considerably, mainly for proper RTF support. This involved a much longer testing and refining cycle.

Posted Thursday, December 24, 2009

EasyCOM2INC 2.11

Changes in EasyCOM2INC 2.11 include: Generated classes inherit from the base class EasyCOMClass (ecombase.inc / ecombase.clw); Changes to when the error message in the class method is generated; Fixed methods that use parameters such as SAFEARRAY; Fixed OleSafeArrayClass when working with multi-dimensional SAFEARRAYs. This is a free upgrade to all customers who have a current (valid) subscription plan. Download EasyCOM2INC and try it before you buy. Price: \$189.

Posted Thursday, December 24, 2009

RPM/AFE Introductory Subscription Prices End Dec 31

Introductory subscription prices for RPM and AFE increase at midnight, December 31 2009. New subscription, RPM Only - US \$168.00; New subscription, AFE Only - US\$239.00; New subscription, everything: US\$271.00. Renewal, and current user enrollment, are suitably lower. Enrollment includes base product for current Clarion versions, product support and new releases for one year.

Posted Friday, December 18, 2009

Updated RPM 7 and 6.3 Installs

Updated RPM 7 and 6.3 installs are available. These RPM updates include a fix to a bug that potentially could have affected drill down reports in C7. Work will continue shortly on RPM support for Fomin Report Builder and ReportDAT. Also a new AFE template and library install is available for C7.0. As with all AFE library installs no server upgrade is required if you're using the most recent server. Servers remain backward compatible and independent of Clarion versions. A new server installer, using the latest FaxMan libraries and SetupBuilder, should be available in January. All current subscribers will have access to this updated server regardless of release date.

Posted Friday, December 18, 2009

iQ-Sync 1.13

iQ-Sync 1.13 adds a command line interface and a MessageBox action.

Posted Friday, December 18, 2009

PropertyGrid Wrapper 1.12

Version 1.12 of the PropertyGrid Wrapper template is available. The new version has been uploaded to members area and a new demo example is available. Modifications include: Enum attributes added to Item Type definition; Moveable Splitter Bar option added; Default position of Vertical Splitter added; New method added - AllowMoveableSplitter ; New method added - GetSplitterPos ; New method added - SetSplitterPos.

Posted Friday, December 18, 2009

Clarion Magazine

The Future of Clarion, Part 2

by Dave Harms

Published 2010-01-18

In Part 1 of this series I compared the development of the Clarion language with the development of the mainstream C and C# languages. I showed that although Clarion is a much more expressive language than C, and far easier to use for Windows development, that advantage almost completely disappears when you compare Clarion# with C# (and, for that matter, VB.NET). I also talked briefly about the .NET AppGen (under development); it's difficult to say anything meaningful about the new AppGen since I haven't seen it, but I'll come back to that topic a little later on anyway.

In this second part I'll take a look at the current state of .NET application development, particularly as it relates to user interfaces, and in the final installment I'll evaluate SoftVelocity's stated strategy for .NET in light of all of this information.

Win32, WinForms and GDI

All four of the Windows apps I showed in Part 1 share one thing: they all create the user interface using a Windows API called the GDI, or Graphics Device Interface. As a result all of these apps have more or less the same look and feel. Figure 1 shows the Win32 version of the sample application, and Figure 2 shows the WinForms version. There's not a lot to choose between them.

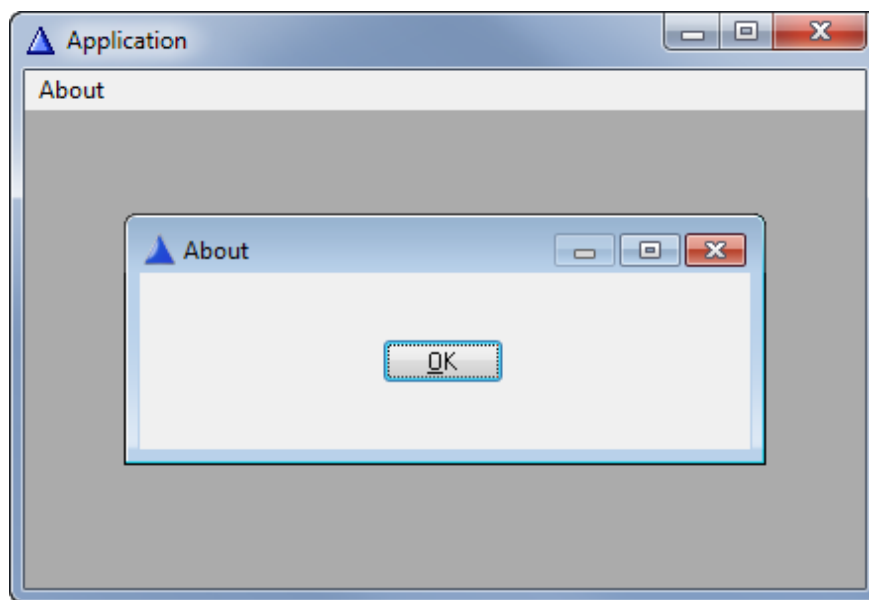


Figure 1. A Win32 application

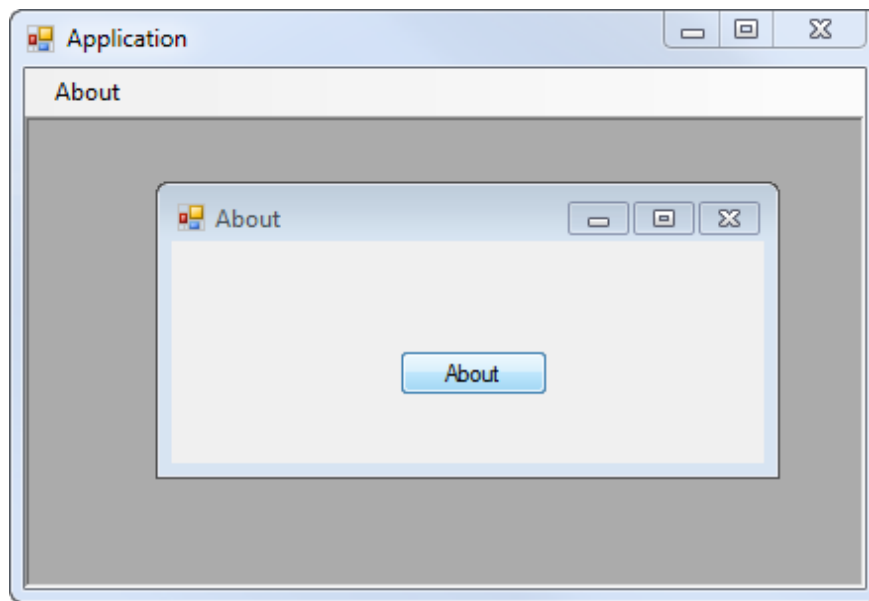


Figure 2. A WinForms application

WinForms apps, of course, have the advantage of a very rich set of controls (as compared to Clarion Win32). But the *way* those controls are drawn is the same, because both use the same API.

GDI is, to my knowledge, as old as Windows, which puts it near the 30 year mark. It's been upgraded along the way (recent versions of Windows use GDI+), but it's fundamentally old technology, and not completely up to the graphic-intensive requirements of a modern user interface.

Wait, *graphic-intensive*? Since when are Clarion business apps graphic intensive? We do just fine with our boring old user interfaces, right?

It all depends on your situation. I still see DOS apps in use at local businesses and institutions. That's a *really* dated user interface (UI). But could you *sell* a DOS user interface right now? Not likely. Windows (with GDI) is the minimum for entry in a good chunk of the world.

The point is, standards change. Expectations change. And interestingly, GDI is being replaced by a technology that doesn't just alter the look and feel of applications, it also fosters a very *different* kind of application architecture.

Every now and then there's a sea change in how users interact with computers. When Windows reached version 3.1, it was pretty much all over for DOS development. And we may well be on the verge of a similar shift towards richer, more graphically intensive interfaces. That seems like an incremental change. It really isn't. This new technology can run on the desktop, on mobile device *and* in the web browser, and in the latter case on multiple platforms (including Mac and Linux).

Are you ready for WPF?

After GDI, WPF

.NET 3.0 introduced a new API called [Windows Presentation Foundation](#), or WPF (originally codenamed Avalon).

There are a lot of technical differences between GDI and WPF, but the most obvious one is the drawing model for creating the user interface. GDI takes a [raster-based](#) approach, WPF a [vector-based](#) approach.

Let's say you need to put a button on screen. GDI draws controls as bitmapped images. WPF, however, draws controls as vectors (lines, curves, shapes etc). Of course ultimately the image on your screen is going to be a bitmap, but WPF

doesn't care about that. It's built on top of DirectX, so it's able to work at a higher level of abstraction and let DirectX (which can take advantage of hardware graphics acceleration) render the images. WPF user interface elements can easily be resized, scaled, rotated etc. (definitely a help for mobile devices).

WPF has direct support for video and audio, both of which are missing from GDI+. Event handling is also somewhat different in WPF.

Unlike GDI, WPF is not limited to the Windows desktop: you may have heard of Silverlight, which is WPF for the browser. Silverlight is a subset of WPF and is available for [Windows](#), [Windows Mobile](#), [the Mac OS](#) and [Linux](#) (the latter via the [Moonlight](#) project); version 4 of Silverlight will also run on the [iPhone](#).

The new rendering engine, which changes the look and feel of the UI, is the biggest change your customers will notice. But for developers, the biggest change is the way the UI is declared in code.

Writing WPF applications is quite different from writing GDI applications, and actually reminds me a wee bit of Clarion Windows. In C Win32 apps, and in .NET WinForms apps, you write code to create the window. In Clarion Win32 apps, you have a separate structure that contains the window definition.

Similarly, WPF graphical elements such as windows and controls are contained in XAML documents, which as you might guess from the name is a particular kind of XML. Like Clarion's WINDOW structure, you can work with XAML documents completely apart from the source code.

The WPF version

Let's have a look at the C# WPF version of the app I showed in Part 1. There are three sets of files, one for the app, one for the main window, and one for the About window.

The App.xaml file is pretty short - the only unique information is the StartupUri setting, which points to the MainWindow.xaml file:

```
<Application x:Class="WpfApplication2.App"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    StartupUri="MainWindow.xaml">
  <Application.Resources>

  </Application.Resources>
</Application>
```

The App.xaml.cs has even less of interest, at this point (I've stripped out the using statements):

```
namespace MyWpfApplication
{
    public partial class App : Application
    {
    }
}
```

MainWindow.xaml contains the main window definition (loosely analogous to a Window structure) including a menu with the About item:

```
<Window x:Class="MyWpfApplication.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="MainWindow" Height="350" Width="525">
<Grid HorizontalAlignment="Left">
    <Menu Height="23" HorizontalAlignment="Left"
        Name="menu1" VerticalAlignment="Top" Width="503">
        <MenuItem Header="About" Click="MenuItem_Click" />
    </Menu>
</Grid>
</Window>
```

And here's the MainWindow.xaml.cs file (again, minus the using statements) with a MenuItem_Click method corresponding to the Click attribute in the XAML file:

```
namespace MyWpfApplication
{
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }

        private void MenuItem_Click(object sender, RoutedEventArgs e)
        {
            About about = new About();
            about.Show();
        }
    }
}
```

The About.xaml file also contains a window definition, but this one has a Close button instead of a menu:

```
<Window x:Class="MyWpfApplication.About"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="About" Height="141" Width="200">
<Grid>
```

```

<Button Content="Close" Height="23" HorizontalAlignment="Left"
Margin="53,43,0,0" Name="button1" VerticalAlignment="Top"
Width="75" Click="button1_Click" />
</Grid>
</Window>

```

And here's the About.xaml.cs file:

```

namespace MyWpfApplication
{
    public partial class About : Window
    {
        public About()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, RoutedEventArgs e)
        {
            Close();
        }
    }
}

```

That's pretty much it for the WPF application. Figure 3 shows the application with the About window activated.

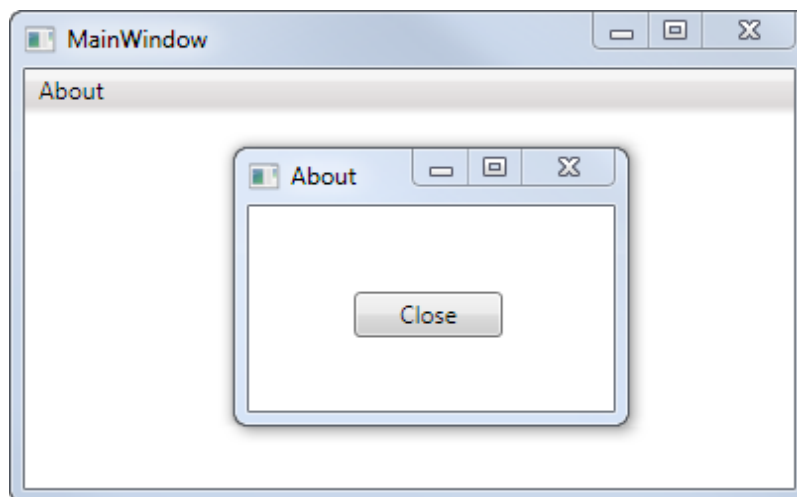


Figure 3. A WPF application

There are a couple of important differences between the WPF app and the WinForms app which go well beyond the scope of this article, but which have important consequences for Clarion developers looking at WPF.

First of all, there's no more MDI. Microsoft has been trying to move away from (some would say kill off) MDI for quite a

few years now so there is no formal MDI support in WPF, although there are a few third party projects that allow you to create MDI-like applications in WPF.

No MDI means you don't have an application frame that can clip child windows so they don't appear outside the frame.

Second, Clarion developers have long used `START()` to create multi-threaded user interfaces. The default in WPF is to have a single user interface thread - all windows in an app are on that thread. You can create a multi-threaded user interface, but it takes extra work. It's extra work in Win32 as well, but Clarion has always hidden that complexity from developers.

Actually the default threading approach in WPF appears to be functionally the same as multi-threading in Clarion prior to C6, since C6 was the first release to have pre-emptive threading. Before that only one Clarion thread could be active at a single time, so although we had "true" threads they didn't work like true threads.

Separating the UI

I said that XAML is in some ways like the Clarion Window structure. Both let you work on the appearance of the window without actually having to write any code. More formally, both are [declarative](#) (you get big buzzword points these days for anything declarative) rather than [imperative](#).

Arguably, both XAML and the Window structure are really window markup languages, just as HTML is a document markup language.

If you compare WPF to WinForms at some point you'll notice that this declarative approach to the user interface is part of a larger change in the architecture of Windows applications.

Note: Unless stated otherwise, everything I say from here on about WPF also applies to Silverlight.

In a WPF app the user interface is treated as something that can stand on its own, and that interacts with the application logic but isn't dependent on the application logic. Similarly, the application logic can interact with the user interface, but isn't dependent on it. The application logic can be isolated, tested (yes, think unit testing), and reused.

If you have any potential for more than one use of your application's business logic (and even if you don't), there are huge advantages to separating that logic from the user interface. This is the drum I've been beating in the ongoing series titled [The Problem With Embeds](#). As much as possible you want to be able to create one code base that, as much as possible, can be reused on desktop apps, in web apps, even in mobile apps. The user interfaces will necessarily change because the devices change; you want a UI that's appropriate to the device.

So step one is to isolate the UI from the application logic (and, of course, the back end datastore). That has benefits for reusability, testability, etc. etc.

It also opens up some fascinating new opportunities for cross-platform deployment. Instead of creating a desktop application that runs only on Windows, you have the option of creating a web application where the Silverlight client app runs on Windows, Linux and the Mac, and on desktop and mobile devices.

Contrast this with the current state of Clarion development. I regularly see Clarion developers post questions in the newsgroups asking how they can port their Clarion applications to the web. The short answer, in most cases, is you can't. I'm not talking about thin client solutions like ClarionNET (not to be confused with Clarion.NET). I'm talking about web applications you can use in a browser. It's just really hard to create a standard Clarion application where any significant portion of that code can be used for something other than the desktop. There are web options for Clarion such as NetTalk, but these don't change your existing applications, they give you tools to create new, web-specific applications.

Although Clarion's Window structure suggests that the user interface (which is a desktop user interface) is separable from the application logic, in fact there is no mechanism in Clarion to accomplish this. Unlike WPF, the Window structure is really just a convenience, and the application logic of most Clarion apps is very closely bound to the user interface. The situation is only made worse when large portions of business logic are dumped into embed points.

In fact, modern application architecture tends to separate not just the business logic from the user interface, but also the business logic from the back end database. Again, Clarion has a semblance of this in its database driver system. But the reality is that a typical Clarion procedure (such as a browse) contains the UI code, the application logic and the back end database access, all tightly coupled. It's very hard to reuse just the data access part of the browse (the view), or just the browse logic (the BrowseClass), or just a UI component (the list box). Instead, if you want to reuse any one part of your browse procedure you end up recreating the code in another form.

Layers and application architecture

Application architecture is really just the overall design of the application, showing how the major functional aspects interact. Traditionally, the only architectural terminology applied to Clarion applications is "client-server". There are really only two parts to this architecture: that's right, the client (the Clarion application) and the server (the database server). And if you're doing TPS access instead of SQL access, you really don't have a server component.

We don't often think of Clarion applications as having an internal architecture. And that's because there isn't one, in a formal sense. Clarion applications are just collections of procedures, and if they're ABC apps they also make use of a lot of standard classes in those procedures. But all of this code, the business logic, the user interface, the data access, in any given procedure it's just all tossed together in a general mishmash.

Clarion applications are an example of what Eric Evans, in his landmark book Domain Driven Design, calls "The Smart UI 'Anti-Pattern'". Although Evans tags this as an anti-pattern (a bad thing) he does say that it's legitimate in some contexts.

A few of Smart UI's advantages:

- High productivity for simple apps
- Requires less developer training
- Relational databases work well
- 4GL tools work well (*leaving aside the question of whether Clarion is a 4GL or a 3GL*)
- Applications are isolated from each other, so changes to one piece of code don't necessarily affect other programs

A few of Smart UI's disadvantages:

- Applications are isolated from each other - integration is only via the database
- There is no abstraction of the business logic, and no reuse of behavior
- Refactoring is difficult
- Complexity is a problem; you tend to create more programs, not richer programs

The alternative to the Smart UI (traditional Clarion) model is the layered application model, where each layer has a specific responsibility. As noted above, the user interface layer is separate from the business logic, which means that you can reuse your business logic for different interfaces. That can be a hard concept to get around, given that we're all so used to creating business logic that directly interacts with the screen (see [The Problem With Embeds, Part 4: The CalcValues Code](#)).

Does it have to be this complex?

Isn't Clarion the perfect example of how Smart UI can be a good approach? Clarion developers create enormous applications that work. Is that a bad thing? Why make application development more complex than it needs to be?

You may be tempted to think that this is just another example of Microsoft making software development more complicated than it has to be. And if you think that, you're wrong. You can blame people if you like, but not one company. Software is simply becoming more complex everywhere.

In other words, this isn't about Microsoft. This is about the evolution of software development.

The short answer is that to create code that's reusable across multiple platforms and multiple applications, you need an architecture in which the application is constructed out of more or less discrete pieces that work together to create something that's greater than the sum of its parts. Think of your applications as ecosystems of objects working together.

That's not the Smart UI, nor is it the Clarion architecture. Clarion as we know it works because it makes it easy to reproduce things. You can crank out new procedures quickly because Clarion writes all that code for you. And yes, if you're on ABC then it's generating a bunch of object-oriented code. Does that mean you could theoretically use Clarion to take full advantage of WPF and Silverlight?

The future Clarion WPF architecture?

Let's say you've somehow been talked into taking this crazy, modern approach to software design. As a Clarion developer, where do you start? Well, you don't. This kind of software design is almost diametrically opposed to the structure of the typical Clarion application.

I mean, they're just not close *at all*.

There's nothing about templates per se that precludes creating a layered architecture with a high degree of reusability. But there are significant barriers in the way Clarion, at present, does applications.

ABC's own design doesn't make it easy to reuse code. How often have you thought it would be nice to take a browse from one window and reuse it on another? I've wished often for that. But it's pretty much impossible because of the design of ABC and the limitations of the AppGen. The Clarion templates don't generate disentangled classes; they generate highly entangled classes. The UI code is all mixed up with the business logic which is all mixed up with the database access.

But it's not just the code that's entangled. The whole Clarion AppGen concept is built around the idea of a screen designer, and being able to insert your own code in the context of a screen designer.

That's just not how WPF works. The whole idea is you can treat the UI layer as a separate thing. In fact, you can even give the UI layer to a designer to work with, and that's cool because it doesn't contain your business logic anyway.

But there's another problem with WPF when it comes to Clarion.NET. The SharpDevelop IDE doesn't have a WPF designer. It used to have one, but in order to display WPF you need to parse it, which meant someone had to write a XAML parser, which essentially means reverse engineering how Microsoft parses XAML. It's a lot of work, but someone had taken a stab at it and contributed it to the SharpDevelop product. Then Microsoft announced they would be releasing their XAML parser, and so the SharpDevelop folks pulled theirs from the product, because why keep trying to build/fix something complex like a XAML parser when you can just get it from Microsoft?

At some point SharpDevelop will get its XAML parser, and a WPF designer, and then at some point after that, presumably, Clarion.NET will get that feature as well.

At that point, will Clarion be ready for the world of WPF? Maybe. If you really want to do the work you can actually create WPF applications already, although it's a clunky, code-only approach that looks more like WinForms development.

And we don't use Clarion to slow down our productivity; we use it to accelerate past our competitors.

So what's in the cards for Clarion.NET? What technologies will be supported? WinForms? WebForms? WPF/Silverlight? What will the application architecture look like? In the [third and final installment](#) I'll take a closer look at SoftVelocity's announced plans, and I'll wrap up with my conclusions.

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

Reader Comments

Posted on Thursday, January 21, 2010 by David Bratovich

Dave,

I agree that...

```
<Page xmlns='http://schemas.microsoft.com/winfx/2006/xaml/presentation' xmlns:x='http://schemas.microsoft.com/winfx/2006/xaml'>
  <TextBlock Cursor="Hand" VerticalAlignment="Center" HorizontalAlignment="Center">"XMAL is the future"</TextBlock>
</Page>
```

I currently use XAML in CW6/7.1 using the CodeJock Markup label control. It is a subset of the SilverLight feature set but there is enough there to start learning how to control and handle the syntax. It currently lacks serious data collection controls and 3D transformation methods but CJ keeps adding more to it every release.

The future looks exciting!

Dave Bratovich

Posted on Thursday, January 21, 2010 by David Bratovich

Dave,

I agree that...

```
<Page xmlns='http://schemas.microsoft.com/winfx/2006/xaml/presentation' xmlns:x='http://schemas.microsoft.com/winfx/2006/xaml'>
  <TextBlock Cursor="Hand" VerticalAlignment="Center" HorizontalAlignment="Center">"XMAL is the future"</TextBlock>
</Page>
```

I currently use XAML in CW6/7.1 using the CodeJock Markup label control. It is a subset of the SilverLight feature set but there is enough there to start learning how to control and handle the syntax. It currently lacks serious data collection controls and 3D transformation methods but CJ keeps adding more to it every release.

The future looks exciting!

Dave Bratovich

[Add a comment](#)

Clarion Magazine

The Future of Clarion, Part 3

by Dave Harms

Published 2010-01-19

In Part 1 of this series I examined the evolution of the Clarion language, showing how its tremendous advantage in WinAPI application development over the C language largely evaporates when you compare Clarion# and C#. In Part 2 I explored WPF/Silverlight, Microsoft's new user interface API for .NET, and I discussed in very general terms the current state of architecture for WPF and Silverlight applications.

In this final installment I'll look at SoftVelocity's stated plans and presumed development resources, and I'll draw whatever conclusions are possible about the future of Clarion.NET development.

The stated architecture

At the 2009 Australian DevCon, Bob Zaunere laid out SoftVelocity's [planned application architecture for .NET](#).

Design goals for the generated code include:

- All of the business rules and business validation logic will be generated in a domain model tier and not within the UI tier
- The data access tier uses LINQ via SV's LINQToFileProvider, LINQToSQL, and future LINQ providers
- The data access tier can use the managed .NET IP driver
- The LINQToFileProvider incorporates an ABC-style framework - think of it as the best of ABC on top of LINQ
- LINQ to SQL is itself a lightweight ORM; the templates are being designed so that in the future they could generate using other heavier ORMs such as Entity Framework, NHibernate, CSLA etc

These goals include a clear focus on a tiered application architecture (a tier isn't always the same thing as a layer, but I think

it's safe to use these terms interchangeably here). There is no mention of WPF at least for the initial templates. That's to be expected since WPF support needs to be integrated into the SharpDevelop code base first. So all of the UI code on the desktop will be WinForms code, and the mobile code will be Windows Mobile, and the web code will be WebForms (ASP.NET).

Unfortunately, all three of these technologies are looking a bit second rate these days. WPF is gaining mindshare at the expense of WinForms, Windows Mobile is losing market share to the iPhone and is expected to take another hit from Google Android, and ASP.NET MVC is reportedly overtaking ASP.NET for new development.

The data access layer is based on LINQ, which is an increasingly popular technology for querying databases (and XML as well as collections of objects). And there's a nod to the major object-relational mapping (ORM) players. But again, we're not talking mainstream, actively developed technologies, at least in the first iteration. LINQToSQL is a Microsoft product, but Microsoft has [stated](#) that Entity Framework, not LINQToSQL, is now the recommended way to do data access.

The reasoning behind the LINQToFileProvider seems clear - provide access to the existing file drivers, in particular the TopSpeed driver. One fly in the ointment here is it appears the file driver code is still Win32, so you can't create a 64 bit .NET app that uses the file drivers. The downside is that this another chunk of code that SoftVelocity has had to write, and which will have to be thoroughly debugged.

And it seems that SV is going down the same path Microsoft took with LINQToSQL; they're trying to apply a great querying technology to a task that goes beyond querying. Microsoft's solution: Create the Entity Framework.

SoftVelocity's solution: Create a LINQ provider that embodies ABC's FileManager and RelationManager behavior.

LINQ uses an SQL-like syntax, so if you've been thinking of brushing up on your SQL now you have another reason.

A difficult choice

SoftVelocity has been clear: Clarion.NET is the future of the Clarion product line. And that future demands new users.

This presents something of a dilemma.

Major technology changes (and make no mistake, going from Win32 to .NET is every bit as major as going from DOS to Windows) are always disruptive, and they inevitably cause some Clarion devs to take a pass.

So if you're SV, who do you cater to? Your existing customers who want and need a reasonable upgrade path to .NET? Or do you go after the new market, and forget about an upgrade path (and probably about Clarion# too) and focus your resources on bringing in revenue from C# and VB.NET developers? Ah, but that's tricky too, because not many of those folks are going to be willing to give up Visual Studio for a modified SharpDevelop IDE.

The focus is pretty clearly on getting Clarion folks moved over to .NET. Besides the IDE disincentive, the dependency on a proprietary runtime from a small vendor, an untested LINQ provider, a second tier provider in LINQToSQL and the lack of a clear WPF development path all mitigate against a VB.NET or C# developer switching to Clarion.NET (even if you give them templates in the language of their choice).

But it's going to be a challenge for existing Clarion devs too. They're going to be moving from an everything-in-one-bucket architecture to a tiered architecture, so things won't be where they used to be. Chances are they'll need to get at least a passing familiarity with the .NET framework if they want to write embed code. And going by SV's announced plans, they'll be creating old-style GDI applications, old-style WebForms applications, and old-style mobile applications. Not that there's anything wrong with that.

Once those Clarion folk who want to upgrade have upgraded, where does the growth come from?

It seems to me there are two markets: Visual Studio developers looking for code generation tools to enhance their productivity, and would-be developers who just want to get the job done.

I don't know which market SoftVelocity is targeting, but I think it's safe to say that if they want to reach the Visual Studio crowd they need two things: WPF/Silverlight support, and a Visual Studio addin version of the AppGen.

When does AppGen get here?

The big question on the minds of many Clarion devs is when can we expect to see a usable Clarion.NET AppGen? Usable doesn't just mean it can generate some code. Usable means it generates usable code.

The .NET templates are brand new; in fact SoftVelocity appears to be talking about three template sets. That's a lot of code to test and to get solid, along with whatever additional support is needed in the language and the runtime. It's not likely going to be a quick process even once the AppGen is there.

We're using a Win32 product that's highly mature. How long will it take to achieve that same level of maturity in .NET? Years, I should think.

Back when Clarion Software merged with JPI (LDC excepted) we knew who the product developers were. We met them at conferences. We (well, some of us) interacted with them online. And there were some brilliant folks there.

From Russia with love

Who is building Clarion.NET? Alexey Solovyev appears to still be involved. Scott Ferrett occasionally pops into the newsgroups to say a word. But where is the rest of the team?

Apparently they're in Russia, according to a tip I received. The outsourcing company Arcadia [lists SoftVelocity](#) as one of its clients. [Another page](#) describes one of Arcadia's projects, without naming the client. The project title is "Implementation of existing compiler and IDE on .NET platform" and the description includes this text:

Arcadia's customer, who is the developer of the programming language and IDE, decided to implement it on .NET platform to extend it so that to take full advantage of the .NET Framework. The new compiler will be integrated with the .NET Framework, making all .NET Framework 2.0 libraries easily available to programmers, while maintaining compatibility with the syntax and features of the existing programming language. This will enable programmers to use their knowledge to build and deploy .Net applications for all of the platforms supported by .NET (Desktop, Web, and Mobile), including XML Web services and ASP.NET pages, and to integrate that code with code written in other .NET languages.

To take advantage of full .NET platform power the existing programming language was redesigned and new language was developed. There were two main tasks:

- New programming language compiler implementation;
- New programming language IDE implementation as plug-in for existing IDE.

Technologies used specifically mention SharpDevelop 2.0. The project started in 2004, and is stated as taking 500 person-months over a 54 month period. That's a team of roughly nine people.

I suppose Arcadia could have another client needing this kind of work, but it seems pretty unlikely.

Outsourcing the new IDE isn't necessarily a problem - after all, Clarion Software initially outsourced work on the Windows product and that turned out pretty well. Then again, the LDC team members were identified directly with Clarion and had a reputation as innovators; presumably they had a strong motivation to come up with some pretty cool stuff. Is an outsourced team as likely to produce something new and impressive, or are they just going to code to spec? Time zone differences, cultural differences, and language barriers may also have taken their toll on the C7/Clarion.NET development process. Is outsourcing a contributing factor to the late delivery of C7?

And what does this say about future support for WPF and Silverlight, assuming that outsourcing is the approach going forward?

Arcadia's list of [technologies used](#) does include Silverlight, but that's the only page where I could find it listed and there's

no mention of WPF anywhere. A brief perusal of their [case studies](#) indicates much of their work is done in Visual Studio 2005. Is this a company with the resources to build a WPF/Silverlight-capable AppGen? Will SoftVelocity go elsewhere for that expertise, or does it already have this in-house? And is a WPF-enabled AppGen all that practical in the first place, or are conceptual differences between WPF development and the AppGen just too great?

Can you bank on Clarion?

There are some unhappy Clarion campers around these days. Maybe you've noticed. Maybe you're one of them. Or maybe you're cool with all things Clarion, and you wonder what the fuss is about.

Truth be told there have always been whiners and complainers in the Clarion community, as there are in just about any community. Bruce Johnson points out a [quote by Bjarne Stroustrup](#), inventor of C++: "There are only two kinds of languages: the ones people complain about and the ones nobody uses."

Some folks like the attention complaining brings them. Others feel their concerns with SoftVelocity (and before them, TopSpeed and Clarion Software) have fallen on deaf ears, and so in desperation they take things public. A few are just jerks.

On the other hand, there really are some things to be concerned about, including the quality of the product and the timeliness of delivery.

But how about a little historical perspective, just one more time? Was ClarionLand once a more hospitable place? Were there good old days, and are these days really that dark?

There have been bad times before. In fact, the history of Clarion is about disasters and near-disasters as much as it is about roaring or even modest successes:

- Clarion 1.0 came out with a dongle, at a time when public sentiment was turning rabidly against copy protection. That almost killed product.
- CPD was a winner (and arguably the pinnacle of Clarion's market share).
- CDD (Clarion 3) was a fiasco of legendary proportions that almost put Clarion in the grave again.
- Clarion for Windows was another good call, thanks largely to the bright minds at JPI.
- The end of the millennium brought what some of us still think of (and not fondly) as the Frank Watts era. Watts was [brought in](#) to "lead the charge in U.S. and global markets for the company's Clarion software product line." Instead he presided over the dismantling of the company into SoftVelocity, which took over the Clarion product line, and Sensium, which focused

on consulting and, headed by president Arthur Barrington (Bruce's son), looked to cash in on the [dot com boom](#). In the process businessman Hank Asher picked up the big brains from the London Development Center (LDC), leaving SoftVelocity with some limited options to use these folks on a contract basis. Clarion was on life support again.

- SoftVelocity made some notable (if not terribly dramatic) improvements in Clarion during the [noughties](#). Clarion was breathing without a ventilator. Yay.
- Clarion 7.x has, to date, been something of a disappointment. It's long, long overdue and as of January 15 2010 is still too buggy for many developers to use comfortably.
- There is as yet no Clarion.NET AppGen (although it has been announced for 2010). The case for the Clarion# language remains unclear. Clarion as a full-fledged .NET code generation product is still being formulated, and no one seems to know exactly what it's going to look like when all the pieces are there, or whether it will eventually have full support for WPF and Silverlight.

What conclusions can be drawn from all this?

The **first** point that comes to my mind is that you can't look back and say how much better it was in the good old days unless you're pretty selective. There have been good days and there have even been glory days (the heyday of CPD) and there have been bad days.

The **second** point is that SoftVelocity, beyond all reasonable expectation, is still here, and still producing Clarion. You gotta give these folks full marks for persistence.

The **third** point (and the one I often console myself with) is that there was time when Clarion was a rock star. Not a Mick Jagger kind of a rock star, more like a really good indie rock star that no one in the media ever talked about. A lot of us got into Clarion back in the CPD days. Those really were pretty good days. Well, as long as you were on the right build of Clarion. Whatever.

The **fourth** point (this may go on for a while) is that we've all put up with a lot of crap over the years, both from the keepers of Clarion and from the ignoramii who mocked our toolset. Yes, we've dealt with more crap than most developers, but we did it because of the immense productivity offered by the templates and the AppGen (and before that, the Designer). But raw language productivity has improved a ton, and the Clarion advantage, while still potentially huge, isn't what it once was.

The **fifth** point is that it's lunacy to base your business plans on SoftVelocity's projected release dates. C7 is years late. I'll be pleasantly shocked if there's a .NET AppGen this year. And how long will it take for the template sets to mature? Make your decisions based on what you have in hand, not on what may or may not appear down the road.

The **sixth** point, which isn't obvious from the above list but I'm on a roll so what the heck, is that there really isn't a direct competitor to Clarion out there in the big wide world. Not on Win32, and not on .NET, at least not to my knowledge. There are a couple of .NET products that show some promise, like Genwise Software Factory and Sculpture, but they're also different from Clarion in important ways. So yeah, SoftVelocity could absolutely clean up. If they're in time.

The **seventh** point, which follows on the sixth, is that SoftVelocity is unlikely to conquer the world as long as it's riding the SharpDevelop horse. The IDE wars are over, and Microsoft has won. Visual Studio isn't just an IDE, it's an entire development ecosystem. No one out there's going to give a fig for Clarion#. They will care about a code generation tool that plugs into Visual Studio and generates C# or VB.NET or whatever. But maybe there's enough of a market among the "just get it done, I don't care if it's not WPF/Silverlight" crowd.

Number **eight**: When it comes to Win32, you gotta be nuts to switch from Clarion to something else at this stage in your career, and with all that existing Clarion code in hand. Even if SV were to fold its tent tonight (and I strongly suspect the company will be around for years yet) you could still use 6.3 and maybe even 7.whatever 'till you're ready to retire to your villa on the French Riviera.

Number **nine**: Migration from Win32 to WinForms is a possibility, but you know it's gonna hurt. Probably a lot.

Number **ten**: Migration from Win32 to WPF is gonna hurt more than Win32 to WinForms.

Number **eleven**: No pain, no gain.

Number **twelve**: You can ride Win32 to retirement, or you can suck it up and start learning .NET. You need to know VB.NET or C# or both. Learn them before you learn Clarion#. It'll be easier that way because there are tons of resources for learning those languages. If you know either one you'll be able to write Clarion# code just fine.

Number **thirteen** (no, I'm not superstitious): Absolutely get Visual Studio, even if it's just the Express version. Yes, it'll make you moan and complain about the Clarion.NET IDE. That's inevitable. But you're shortchanging yourself in the .NET world if you don't spend some time with this puppy. You might as well start with VS 2010. It rocks.

Number **fourteen**: We're cranky, and they're not going to take it. Well over a year ago I suggested privately to someone at SV that the average Clarion developer was a bit more frowny-faced than the company seemed to realize. I was told that I needed to have my **FUD** removed. Heck, I'd love to have my FUD removed. I just don't know how that's supposed to happen.

All of this good cheer came back to mind this week when a thread I was participating in was summarily deleted from the C7 newsgroup. Robert Paresi started the thread, titled SV - Hotfixes? at 8:17 a.m. on January 14. He wrote of his frustration in seeing bugs fixed in the PTSS, but not being able to get access for long periods of time. That kicked off a fifty-plus message discussion that raised questions such as:

- Why aren't builds released more frequently?
- How often is too often? Is nightly way too much?
- Is the C7 build process fully automated?
- Would nightly builds encourage feature creep?
- Would too-frequent builds cause too much work for third party vendors?
- Would devs pay for interim builds, and would charging extra for them be a good idea?

I thought it was a pretty good discussion, although I confess to saying one or two disparaging things about the C7 quality control process. I really didn't think the thread deserved deletion. But gone it was, and it's not the first time SV has killed a thread. Of course that's their right; it's their server. But this kind of stuff has a bad smell.

Number **fifteen**: Don't expect SV to change. I know I can't make them engage with their customers, or participate actively in difficult discussions, or try to understand why there's as much discontent as there is right now. It would be nice if they did all that. I think it would help a lot. We can all dream.

The grand conclusion

It's relatively easy to see that the migration path from Win32 to .NET, particularly with WPF, is a difficult one. The aging Clarion application architecture isn't going to cut it, which means that a lot of the embedded code out there isn't going to make the trip unchanged, if it makes it at all.

And there's really no doubt that WPF/Silverlight is the way forward on the .NET platform. Microsoft continues to support GDI+, but that's now old technology.

Clarion still does code generation better than any other technology I've seen, although the gap is narrowing. The real issue is this: how will SoftVelocity implement code generation in Clarion.NET? And how/when will the .NET AppGen play with WPF and Silverlight? Unfortunately there's no reliable data to help answer these questions.

Ultimately, I think the message about Clarion and .NET comes down to this:

- If you have to make a decision on your Clarion-related .NET direction now, do it by evaluating your current needs against what exists right now, not on what's promised. If you have to choose now, all you know you have is Clarion# the language. Just don't assume you'll have such and such a product at such and such a time. That's a fool's game with almost any vendor.
- If you don't have to make a decision now, it's easy. Wait until you do have to make a decision, and at that time evaluate

what SoftVelocity has produced.

So stop worrying about SV's house, and get your own house in order. They're going to do what they're going to do. We're here to help you do what you need to do.

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

Reader Comments

Posted on Tuesday, January 19, 2010 by Robert Hutchison

Great article Dave

Bob Foreman stated at the Aussie Dev Con that Soft Velocity reads Clarion Magazine so they will by now have read this article for sure.

Come on Bob or Bob I would like to hear your comments here please.

Posted on Tuesday, January 19, 2010 by Dave Harms

Thanks, Robert.

Posted on Tuesday, January 19, 2010 by Steven Muller

Excellent informative article Dave, I couldn't agree more.

Posted on Tuesday, January 19, 2010 by Djordje Radovanovic

Excellent article.

Djordje

Posted on Wednesday, January 20, 2010 by Mihai Palade

Yes, good article. Thank you!

Posted on Wednesday, January 20, 2010 by Dave Harms

Thanks, everyone. I'm glad you found it helpful.

Dave

Posted on Wednesday, January 20, 2010 by HaCe

It's also absolutely my opinion - thank you for the articles. On the "pro" side for clarion you maybe forgot the power of EVALUATE() we use extensively - but anyway.

It is a very sad story for us since we using clarion for 15 years. I wish David Bayliss were here with us these days...

With Regards,

Harald

Posted on Wednesday, January 20, 2010 by Dave Harms

Harald,

You make a good point about Evaluate(). You can do runtime expression evaluation in .NET, but I don't know how it compares on a feature by feature basis or whether performance is similar.

Dave

Posted on Wednesday, January 20, 2010 by John Farmer

Dave,

Excellent article! Well stated, well thought-out comments. It reminds us that there are pros and cons with any story.

John Farmer

Posted on Wednesday, January 20, 2010 by Dave Harms

Thanks John.

Posted on Thursday, January 21, 2010 by David Penny

Very interesting article, and sums up pretty much my own feelings.

I have been using Clarion since the first Win version, but have almost given up on C7 now - cannot get it to compile my very large legacy applications, and there's not enough time to work out why.

I'm afraid my future plans no longer include Clarion - it's now used purely for maintaining legacy applications.

All new work is done in VS2008 using ASP.NET/MVC or Windows/Silverlight.

As you state, Microsoft has won the war (for the moment).

I would love to continue using Clarion, it has served me very well over a long period of time, and in terms of productivity cannot be matched, but I have a nagging feeling I cannot bet the future of my company on it.

Posted on Thursday, January 21, 2010 by Robert Wagner

The best article on Clarion, ever!

Posted on Thursday, January 21, 2010 by Dave Harms

David,

I'm a big fan of MVC. The new ClarionMag web site (currently in private beta) is written in ASP.NET MVC, and the current site, which is a Java web app, is also an MVC design.

But all apps of any size involve a certain amount of grunt code, MVC included. And that's where code generation is key. In fact, I've written templates to generate a good chunk of the code needed for the new site, using C7. It's not an ideal solution but it's still better than the alternative.

Dave

Posted on Thursday, January 21, 2010 by Dave Harms

Wow, Robert! Thanks!

Dave

Posted on Thursday, January 21, 2010 by Trevor Cocks

Nice article Dave, I'm pleased to see that you're tackling the real issues head-on.

There's been a definite change in tone in our articles over the past few months... more "gritty", and I like it, less "company-line". I hope SV read it as there are underlying messages there they need to take heed of. Do these guys respect anyone's views?

I too believe .Net is the only long-term way forward, and we're late to the party - I just hope SV can catch up.

I particularly agree with point 7 about Clarion as a code-generator "plug-in" to the VS IDE.

I have held this view for many years - "don't fight a war you can't win"...

Had SV gone all-out for .Net with Clarion7 (no win32 offering), used VS as the IDE (not SharpDevelop) - all us win32 developers (and I use the word developers, not programmers) would now know ALOT more about .Net/C#/VB.Net, the product would be more mature/stable, AND we'd probably even have a reasonable Win32>-to-.Net port ready.

In my view anyone who wants to stay Win32 could have stuck with C6.x (with a few SV bug fixes to round it off).

As it is now, we're faced with: do nothing, upgrade twice (c6 > c7 > .Net) or wait probably a VERY long time before Clarion.Net is mature & stable. (by which time the rest of the .Net world has moved on and we're on catch-up again).

Regards

Posted on Thursday, January 21, 2010 by Dave Harms

Trevor,

You're not the only one to suggest that SV would have been better off porting the existing IDE to 32 bits, and then doing something completely new for VS.

Of course, hindsight is 20/20. We don't know how difficult it would have been to create an AppGen addin for, say, VS 2005 (which presumably is when serious work started on the new IDE).

VS 2010 definitely presents better opportunities (by one rumor, the *first* opportunity for AppGen integration). Z has said in the past they've considered the idea.

But I have absolutely no idea whether there be a VS addin. Obviously there's a huge investment in the new IDE.

Dave

Posted on Thursday, January 21, 2010 by Peter Gysegem

If SV does not move to the Visual Studio (VS) platform, they will always be behind Microsoft technology. First, #Develop would have to adopt it, then SV would have

to follow. Each of these steps could take up to a couple of years. Clarion developers would thus be a couple of years behind developers using VS who would be creating state of the art applications with the latest technology.

Clarion developers have made significant investments in both time and money in Clarion and really want SoftVelocity and Clarion to succeed and thrive. Any arguments or discussions about the future directions SV is taking Clarion, no matter how heated they become, are between people who all want the same result, to have a development environment that gives us advantages of productivity, (relative) ease of use, and access to the latest technology.

Unfortunately, the publicly stated direction seems to promise questionable productivity, a still-buggy IDE, and technology that will stay years behind developers using Microsoft's tools (Clarion.Net uses .Net 2.0 while MS is at 4.0). I may be exaggerating somewhat in degree but not in direction.

My plea to SoftVelocity is to give us (me) a convincing reason to stay with Clarion.

Posted on Thursday, January 21, 2010 by Dave Harms

Peter,

Actually Clarion# does have at least partial support for .NET 3.5, but the point is still valid. It takes a whole lot of work to keep up with MS on the IDE/tools front, and it's not going to get any easier going forward.

Dave

Posted on Thursday, January 21, 2010 by Lee Vasic

I've been using Clarion since version 1. I've been through all of the ups and downs and I've made a lot of money using Clarion. There was a time when companies wanted software that worked and didn't ask you what it was written in. Now everyone wants to know if it is written in .Net.

It seems to me that there is a paradigm shift in what SV wants to be. Clarion used to be at the cutting edge of technology. We were able to do stuff that VB, Dbase, Delphi etc, couldn't even dream of. Clarion was awesome!

Now it seems that Clarion wants to be a 3rd party product to Microsoft. If that's where they position themselves they will always be way behind. They will always be waiting for Microsoft or discovering Microsoft bugs&

From my experience Client Server and ISAM files now sound like 16 colors in DOS. The new vision is a web front end with a SQL back end. SQL should stay as pure as possible. Most of the work should be done on the Server anyway.

Microsoft has turned this whole thing into a mess. They change the name or location of something and call it a new technology. I still believe that Template based programming is superior to Object Oriented programming. These black boxes are really the dumbing down of programmers. Pretty soon nobody will know how to write a calendar or numbers to words function.

We need to get back to the basics. There are six procedures; Menu, Table, Form, Report, Process & Source. We place those on a web page and use AJAX for the IO.

I want to write data-centric applications that make money. I need tools that will help me do that. Clarion used to be that tool and I hope it will be in the future, but at this point I can't wait. I need to keep the money rolling in.

Posted on Thursday, January 21, 2010 by david griffiths

Great article but i think that there is to much emphasis on trying to keep up with MS. I don't care about the technology. I just need a tool that makes a great application ,bug free and fast so that i can make some money.

Posted on Thursday, January 21, 2010 by Dave Harms

Lee & David,

You can argue that Clarion became successful exactly because it wasn't a mainstream programming tool. It let people get started building business apps when they didn't have much (or any) software development experience.

In that respect, Clarion is a victim of its own success. CPD was good - let's make it better! CDD was mostly a flop, but Clarion for Windows was another success. And Clarion/Topspeed/SoftVelocity's customer's have constantly pushed for more features, more capability.

Where we're at now, however, is a somewhat uncomfortable and, I think it's safe to say, for SV an enormously expensive middle ground between being a unique and completely alternative dev too and being an adjunct to Visual Studio.

Dave

Posted on Thursday, January 21, 2010 by Alex McCullie

Hi Dave

Thank you for the three discussion articles and it's good to put the issues on the table. I think it is a genuine dilemma for many Clarion developers on how to move forward.

My situation is somewhat different in that most of my development is in .NET via Visual Studio. I develop "smaller" custom web-based database applications for corporates in Australia, where, quite frankly, web-based .NET solutions (intranet and Internet) dominate. Multi-tier architectures are common-place for all sized apps. The reality is that I don't get asked for Windows-based solutions any more.

I'd love to have the app gen capability in the .NET world for web apps. For that reason I took out an initial enterprise subscription on Clarion.NET in the hope that would eventuate. I now ignore the various betas still hoping that some sort of rapid development environment will come along. Right there isn't much difference between VS and Clarion# except that VS is a whole lot more stable and better developed!

Where I have been able to use a rad tool then I have worked with Ironspeed for the simple reason that I can genuinely say to a client that this is fully VB or C# in .NET. I can give them the complete source code without any proprietary dlls. I learnt hard lessons many years ago that in the custom work it is too hard fighting against the dominant software tools. Corporates are not interested in technical niceties of an unknown product over the dominant one. Their primary interest is in support and the supply of potential developers.

Again, Dave, good articles, if somewaht unsettling.

Alex

Posted on Friday, January 22, 2010 by Dave Harms

Thanks Alex.

I agree, web apps are only getting more important, although I expect ASP.NET MVC will take over from WebForms if it already hasn't.

The push toward web-based apps is also evident in the massive interest in Silverlight 4. It'll be interesting to see how this shakes out.

I tend to think that's a greater need in the corporate world (where Clarion has always been a tough sell) for web apps than in the smaller businesses many Clarion devs have as clients. But the shift is well under way.

Dave

Posted on Saturday, January 23, 2010 by Philip Cumpston

Thanks to all contributing to the discussion.

I have written an app in C7.1 that actually appears to be working pretty well. For my own purposes, I would like to be able to convert it to .NET and run it remotely using the internet for access on any web server platform.

Clarion# and .NET should be able to give me that functionality, as well as being able to function as a add-on to Microsoft Visual Studio. The promise of being able to convert standard clarion apps to .NET and hence to mobile telephone and web is tantalising. MS visual studio is unlikely to deliver on that, and I think SoftVelocity should be.

Phil

Posted on Monday, February 01, 2010 by Michael Lawson

Dave

I've not been reading Clarion Mag for awhile, but I think this is one of the best set of articles ever. It took a very even handed approach to expressing the frustrations we all feel about the future of Clarion development.

Mike...

Posted on Wednesday, February 03, 2010 by Mike Petitjean

This was an excellent evaluation of the state of Clarion. I too have used Clarion since the CPD days (although I'm sure I'm not old enough to have done that <g>). By the time Clarion 7 went gold, my company had already made up its mind to simply proceed with the Clarion.NET subscription we had also purchased. I agree with those who believe that the future of applications is in web application development, especially now that smart phones rule the world. I had hopes that Clarion.NET could lead me gently into that world.

I spent several thousand dollars and about two years learning Java and actually built some simple client-server apps with it. All the Java IDEs I tried paled in comparison to what I was used to with Clarion. Code hints just don't replace code generation!

Recently I tried a subscription to CodeCharge Studio to move to the world of generated web applications. This has been a mixed bag and I have not found it to be as reliable and capable as Clarion.

Since I agree with the points in the article that Clarion.NET is not going to be ready anytime soon and that SoftVelocity will always be playing catch-up with Microsoft's .NET framework changes, I turned to PHP as an alternative language for web apps (I started down the C#.NET road but this looked as cumbersome as Java was).

At Christmas time I discovered an Open Source product called Drupal. This is a web development tool which stores the website information in a MySQL database. There are thousands of third party add-ons and themes available for this tool. It is written in PHP and can be adapted to store in a PostgreSQL, MS SQL or Oracle database easily. The user groups are friendly and helpful. It reminds me of the good old days with Clarion. It is designed so that you can put together a website of some sophistication rapidly without understanding PHP, SQL, AJAX, etc.. So you can be productive out of the box and become an expert as your knowledge grows. Again it reminds me of all the advantages I had with Clarion. If you are interested check out <http://www.drupal.org/>.

All that said I am still making a living with Clarion 6.x and would love a miracle from SoftVelocity to allow me to continue to leverage my existing knowledge while moving forward into another "new" computer age.

Posted on Monday, February 08, 2010 by Dave Harms

Philip,

I think there are a lot of high expectations for Clarion.NET which are going to be difficult to meet. Converting existing applications is one of them. On the other hand, once you move to a layered architecture, particularly with WPF/Silverlight for the UI layer, there's the promise of being able to share much more code between desktop, mobile and web versions of your apps. That's not unique to Clarion# - in fact, Clarion# can't easily do WPF and Silverlight yet. The hope is that the AppGen will make all of this easier for developers.

Dave

Posted on Monday, February 08, 2010 by Dave Harms

Thanks, Michael.

Dave

Posted on Monday, February 08, 2010 by Dave Harms

Thanks, Mike. Drupal and Joomla are two excellent and very popular choices for content management systems.

As for C# and web dev, it's not as easy as using an existing CMS but it can be very powerful. Keep your eyes open for the new ClarionMag site, which is an ASP.NET MVC web app. The new site is in private beta and hopefully will go into wider beta in the near future.

Dave

[Add a comment](#)

Clarion Magazine

The Future of Clarion, Part 1

by Dave Harms

Published 2010-01-12

Many questions remain unanswered on the .NET front. When will there be a usable AppGen? The new template language is based on Microsoft's T4, but what will the implementation really look like? How will Clarion.NET support Windows Presentation Foundation and Silverlight, the current state of the art in .NET user interface development? How does the Clarion# language compare to C# and VB.NET? What is Clarion#'s role?

I believe these questions are best understood in historical context; for that reason I'm going start by sketching out the evolution of the Clarion language on the Windows platform, as it compares to Microsoft's mainstream programming languages. In this article I'll compare C with Clarion for Windows (the language), and C# with Clarion#. Next time I'll go on to the ramifications of Windows Presentation Foundation (WPF).

In the beginning

I suppose to be comprehensive I should really go all the way back to DOS applications, and compare Clarion Professional Developer to C for DOS. There are two problems with that. One is that this article is going to be long enough anyway, and the other is that I no longer have a DOS version of Clarion installed.

As with the Windows C example that follows, however, the Clarion code for DOS apps was typically far smaller and more readable than the equivalent C code. If anything the difference on Windows was even more dramatic, so in any case I think it's safe to begin this discussion with the Windows platform.

Windows, the early years

In the early years the main language for writing Windows applications was C. Actually C++ hit the scene around the same time as Clarion for Windows, but Clarion programmers were strictly procedural coders back then so it's probably fairer to compare procedural languages.

From [MSDN](#), here's how you create a Win32 application, with a main window and an About window, in C (if you find your eyes glazing over, well, that's probably just as well):

```
#include <windows.h>
#include "generic.h"
#pragma comment(lib, "user32.lib")
#pragma comment(lib, "gdi32.lib")
```

```
LRESULT CALLBACK MainWndProc( HWND, UINT, WPARAM, LPARAM );
```

```
INT_PTR CALLBACK AboutDlgProc( HWND, UINT, WPARAM, LPARAM );
```

```
HINSTANCE ghInstance;
```

```
int CALLBACK WinMain( HINSTANCE hInstance,  
    HINSTANCE hPrevInstance,  
    LPSTR lpszCmdLine,  
    int nCmdShow )  
{  
    MSG msg;  
    HWND hWnd;  
    BOOL bRet;  
    WNDCLASS wc;  
  
    if( !hPrevInstance )  
    {  
        wc.lpszClassName = TEXT("GenericAppClass");  
        wc.lpfnWndProc = MainWndProc;  
        wc.style = CS_OWNDC | CS_VREDRAW | CS_HREDRAW;  
        wc.hInstance = hInstance;  
        wc.hIcon = LoadIcon( NULL, IDI_APPLICATION );  
        wc.hCursor = LoadCursor( NULL, IDC_ARROW );  
        wc.hbrBackground = (HBRUSH)( COLOR_WINDOW+1 );  
        wc.lpszMenuName = TEXT("GenericAppMenu");  
        wc.cbClsExtra = 0;  
        wc.cbWndExtra = 0;  
  
        RegisterClass( &wc );  
    }  
  
    ghInstance = hInstance;  
  
    hWnd = CreateWindow( TEXT("GenericAppClass"),  
        TEXT("Generic Application"),  
        WS_OVERLAPPEDWINDOW|WS_HSCROLL|WS_VSCROLL,  
        0,  
        0,  
        CW_USEDEFAULT,  
        CW_USEDEFAULT,  
        NULL,
```



```
NULL,  
hInstance,  
NULL  
);
```

```
ShowWindow( hWnd, nCmdShow );
```

```
while( (bRet = GetMessage( &msg, NULL, 0, 0 )) != 0 )  
{  
    if (bRet == -1)  
    {  
        // handle the error and possibly exit  
    }  
    else  
    {  
        TranslateMessage( &msg );  
        DispatchMessage( &msg );  
    }  
}  
  
return (int)msg.wParam;  
}
```

```
LRESULT CALLBACK MainWndProc(
```

```
    HWND hWnd,  
    UINT msg,  
    WPARAM wParam,  
    LPARAM lParam )
```

```
{  
    PAINTSTRUCT ps;  
    HDC hDC;
```

```
    switch( msg )
```

```
{
```

```
    case WM_PAINT:
```

```
        hDC = BeginPaint( hWnd, &ps );
```

```

    TextOut( hDC, 10, 10, TEXT("Hello, Windows!"), 15 );

    EndPaint( hWnd, &ps );
    break;

case WM_COMMAND:
    switch( wParam )
    {
        case IDM_ABOUT:
            DialogBox( ghInstance, TEXT("AboutDlg"), hWnd,
                (DLGPROC) AboutDlgProc );
            break;
    }
    break;

case WM_DESTROY:
    PostQuitMessage( 0 );
    break;

default:
    return( DefWindowProc( hWnd, msg, wParam, lParam ) );
}

return 0;
}

```

```

INT_PTR CALLBACK AboutDlgProc(
    HWND hDlg,
    UINT uMsg,
    WPARAM wParam,
    LPARAM lParam )
{
    switch( uMsg )
    {
        case WM_INITDIALOG:
            return TRUE;
        case WM_COMMAND:
            switch( wParam )
            {

```

```
case IDOK:
    EndDialog( hDlg, TRUE );
    return TRUE;
}
break;
}

return FALSE;
}
```

And here's how you would create a similar program in Clarion for Windows:

PROGRAM

MAP

```
Module('Main.clw')
    Main
End
Module('About.clw')
    About
End
END
```

CODE

```
Main()
! Main.clw
```

```
MEMBER('FrameWithAbout')
```

MAP

```
END
```

```
Main    PROCEDURE
```

```
AppFrame    APPLICATION('Application'),SYSTEM,MAX,AT(.,505,318),|
            CENTER,ICON('WAFRAME.ICO'),STATUS(-1,80,120,45),|
            MASK,FONT('MS Sans Serif', 8),RESIZE
            MENUBAR, USE(?Menubar)
```

```
ITEM('About'), USE(?About)
```

```
END
```

```
END
```

```
Code
```

```
Open(AppFrame)
```

```
Accept
```

```
case Accepted()
```

```
of ?About
```

```
About()
```

```
End
```

```
End
```

```
! About.clw
```

```
MEMBER('FrameWithAbout')
```

```
MAP
```

```
END
```

```
About PROCEDURE
```

```
Window WINDOW('About'), AT(, , 205, 55), FONT('MS Sans Serif', 8), GRAY
```

```
BUTTON('&OK'), AT(81, 20, 41, 14), USE(?OkButton), DEFAULT
```

```
END
```

```
Code
```

```
Open(Window)
```

```
Accept
```

```
Case Accepted()
```

```
Of ?OkButton
```

```
Break
```

```
End
```

```
End
```

It's not hard to see the appeal of Clarion for Windows over C. The Clarion code is far more readable and considerably more expressive. Windows are presented as structures, not as code. It takes about 45 lines of easy-to-follow Clarion code to

get the job done vs. about a hundred lines of somewhat more cryptic C code. Start adding controls and behaviors and database access and the difference in readability and lines of code quickly becomes far more dramatic.

Clarion for Windows (as it was then called) presented some clear advantages to developers moving from DOS to Windows. Other benefits, besides reducing the amount of code, included:

- RTL functions (including queues)
- File access grammar (Set/Next etc)
- File driver system (most databases supported)
- Built-in reporting
- The ability to call the WinAPI directly if needed

The advantage of the Clarion *language* was primarily that any one line of code typically wrapped up a whole bunch of calls to the Windows API, which made using that API a whole lot easier in most situations.

And, of course, you had the dictionary editor and the AppGen and templates, so you didn't even need to write all of that code in the first place. It was a complete no-brainer. You'd have to be a masochist and/or a C fanatic to *not* choose Clarion when it came to writing Windows apps.

Moving on to .NET

Now, fast forward to Clarion# at the start of 2010. How does Clarion# stack up against C#, one of the two primary .NET development languages (the other being VB.NET)?

I've written the above program in both C# and Clarion#. I won't list all the code here, just enough snippets to give you the idea.

First, here's how you launch the main application in Clarion#:

```
PROGRAM
```

```
    NAMESPACE(FrameWithAbout)
```

```
        USING(System)
```

```
        USING(System.Drawing)
```

```
        USING(System.Windows.Forms)
```

```
    CODE
```

```
        Application.EnableVisualStyles()
```

```
        Application.SetCompatibleTextRenderingDefault(false)
```

```
        Application.Run(NEW MainForm())
```

And here's how you do it in C#:

```
using System;
```

```

using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    static class Program
    {
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new MainForm());
        }
    }
}

```

How about the MainForm procedure itself? That's in two files. One describes the appearance of the window, and you can consider it to be the equivalent of the WINDOW structure (which doesn't exist in Clarion#). Here are the first few lines of the Clarion# version:

```

MEMBER("")

NAMESPACE(FrameWithAbout)
USING(System)
USING(System.Drawing)
USING(System.Windows.Forms)

MainForm CLASS(),TYPE,NETCLASS,PARTIAL
InitializeComponent PROCEDURE(),PRIVATE
aboutToolStripMenuItem System.Windows.Forms.ToolStripItem,PRIVATE
menuStrip1 System.Windows.Forms.MenuStrip,PRIVATE
END

MainForm.InitializeComponent PROCEDURE()
CODE
SELF.menuStrip1 = NEW System.Windows.Forms.MenuStrip()
SELF.aboutToolStripMenuItem = NEW System.Windows.Forms.ToolStripItem()
SELF.menuStrip1.SuspendLayout()
SELF.SuspendLayout()

```

```

!
! menuStrip1
!
SELF.menuStrip1.Items.AddRange(
    NEW System.Windows.Forms.ToolStripItem[] {SELF.aboutToolStripMenuItem})
SELF.menuStrip1.Location = NEW System.Drawing.Point(0, 0)
SELF.menuStrip1.Name = 'menuStrip1'
SELF.menuStrip1.Size = NEW System.Drawing.Size(292, 24)
SELF.menuStrip1.TabIndex = 0
SELF.menuStrip1.Text = 'menuStrip1'

SELF.aboutToolStripMenuItem.Name = 'aboutToolStripMenuItem'
SELF.aboutToolStripMenuItem.Size = NEW System.Drawing.Size(52, 20)
SELF.aboutToolStripMenuItem.Text = 'About'
SELF.aboutToolStripMenuItem.Click += SELF.AboutToolStripMenuItem_Click

```

And here's the equivalent C# code:

```

namespace WindowsFormsApplication1
{
    partial class MainForm
    {
        private void InitializeComponent()
        {
            this.menuStrip1 = new System.Windows.Forms.MenuStrip();
            this.aboutToolStripMenuItem =
                new System.Windows.Forms.ToolStripItem();
            this.menuStrip1.SuspendLayout();
            this.SuspendLayout();
            //
            // menuStrip1
            //
            this.menuStrip1.Items.AddRange(
                new System.Windows.Forms.ToolStripItem[] {
                    this.aboutToolStripMenuItem});
            this.menuStrip1.Location = new System.Drawing.Point(0, 0);
            this.menuStrip1.Name = "menuStrip1";
            this.menuStrip1.Size = new System.Drawing.Size(284, 24);
            this.menuStrip1.TabIndex = 0;
            this.menuStrip1.Text = "menuStrip1";

```

```

this.aboutToolStripMenuItem.Name = "aboutToolStripMenuItem";
this.aboutToolStripMenuItem.Size = new System.Drawing.Size(52, 20);
this.aboutToolStripMenuItem.Text = "About";
this.aboutToolStripMenuItem.Click +=
    new System.EventHandler(this.aboutToolStripMenuItem_Click);

```

Here's the code in MainForm that responds to a button-click on the AboutMenu item:

```

MainForm.AboutToolStripMenuItem_Click PROCEDURE(System.Object sender, |
                                     System.EventArgs e)
w About
CODE
w = NEW About()
w.MdiParent = SELF
w.Show()

```

And here's the same functionality in C#:

```

private void aboutToolStripMenuItem_Click(object sender, EventArgs e)
{
    About about = new About();
    about.MdiParent = this;
    about.Show();
}

```

I could go on, but you're probably getting my drift by now. The Clarion# code really doesn't look significantly different from the C# code except for the semicolons, the curly braces and the Data section.

I'm not the first person to note this similarity. Even Bob Zaunere, speaking at the 2009 Australian DevCon in Eden, noted that anyone looking at Clarion#, C# and VB.NET versions of a given application wouldn't see a lot of difference between them.

So what's the difference?

Are there still significant differences between Clarion# and C#? Certainly. For one thing, C# is an object-oriented language, while Clarion# is a hybrid language that lets you mix procedural and OO code. Behind the scenes Clarion# creates procedures as methods; when you call a procedure Clarion# creates an instance of the class and calls the method, but from a programmer's perspective you can still write what looks just like regular procedural code.

I don't consider the ability to write procedural code much of a benefit in this day and age. But Clarion# still does have some utility for Clarion developers. These include:

- The ability to port some non-UI code unchanged to .NET
- The runtime library
- VIEWS
- The file driver system

- The REPORT structure
- QUEUES
- Easy way to call Clarion Win32 procedures (just prototype as usual)

The first point, porting code unchanged, assumes that the code in question is disentangled from the user interface, and isolated from anything else that has changed in Clarion#. This means it cannot use REPLACE on constructors. The OVER and ADDRESS functions are no longer allowed, array indexes are now zero-based rather than one-based, and multi-dimensional arrays are no longer nested.

You should be able to port at least some of your application code to Clarion#, particularly if you're already in the habit of encapsulating that code in classes (although there are subtle changes in how classes are handled).

In general, the runtime library appears to otherwise be more or less implemented, and if I recall correctly, it's also native .NET code which means you can use it with 64 bit .NET apps. But. There's always a "but".

VIEWS appear to be fully supported, but at least some part of the file driver system has not been ported to .NET. If, for instance, you attempt to compile and run the Clarion# People sample app on a 64 bit version of Windows with the Target CPU set to Any Processor, the application will crash at the point where you try to open the People TPS file. If you change the Target CPU to 32-bit Intel Compatible Processor, the application will run properly.

The REPORT structure is there in .NET, as noted above, but it's not something I've tested yet.

Queues are, to me, one of the really nice things about the Clarion language. Yes, you can accomplish the same kind of functionality with .NET with a little work (and .NET adds some niceties like LINQ) but Queues are still pretty cool in my book.

What about the AppGen?

First, as of this writing the Clarion.NET AppGen doesn't exist. But if it did exist, the first thing to keep in mind is that the AppGen would probably no longer have a dependency on *Clarion-specific* functionality. Clarion# creates windows the same way other .NET languages do, and it uses the same visual designer. So any advantage the AppGen provides, it provides for other languages as well as Clarion#. In other words, AppGen won't necessarily be a deciding factor anymore in choosing what language you want to use.

Okay, in one respect the new AppGen could have a dependency on Clarion#. If it allows you to create reports using the Report Designer, that would mean you'd have to use Clarion#, since the REPORT structure is a Clarion-specific feature. But SoftVelocity has stated it plans to support VB.NET/C#, so it's not clear whether there will even be a report designer in the new AppGen, especially with the creation of the new .NET DevExpress-based report writer.

Clarion# - what remains?

Interestingly, a lot of what we think of as the Clarion# language is really just the runtime library (RTL). All those classes that do Clarion stuff for you, if your Clarion# code can use them, then any other language can use them too. I'm told you can access TPS files from C#, and certainly you could write your own wrapper in Clarion# and access TPS data, or, say, use a VIEW, from any other .NET language. Language features like queues might be an exception, but I bet those could be exposed pretty easily if they aren't already.

A lot of the RTL is still very useful. But the RTL is not the language.

For DOS and WinAPI application development, compared to languages like C, the Clarion *language* had a tremendous benefit. It resulted in far fewer lines of code being written. It used easily-understood representations of windows and reports. It had easy-to-use event handling via Accept. And you couldn't get that benefit if you write your apps in C.

Surprisingly, in .NET Clarion's traditional *language* advantages have almost completely disappeared. *It takes basically the same number of lines of code to write an application in Clarion# as in VB.NET or C#.* That's an absolutely stunning change from previous platform upgrades.

From a purely language point of view, there isn't much that Clarion does that C# or VB.NET can't do as well. Partly that's because a couple of things have been lost in Clarion# (like the Window structure and the Accept loop), but it's also that the .NET platform itself is a much more powerful platform in which to build applications, and a typical line of C# code now does a whole lot more than a typical line of C code did.

The dark side of Clarion#

I've used Clarion# quite a bit (though not lately), and I have to say that overall I'm impressed with what SoftVelocity has achieved. You can certainly write applications with Clarion#, and they can be desktop apps, or web apps, or even mobile apps. Presumably the first templates SoftVelocity releases will be Clarion# templates, and that means there will be Clarion# applications out there in the wild at some point, if there aren't already.

In theory it's a lot easier to write a .NET compiler than it is to write a Win32 compiler, because the .NET compiler emits human-readable IL code, and you can compare your compiler's output with, say, the output from Microsoft's compilers for the same kind of code.

The dark side of Clarion# is this: *language development takes resources.*

SoftVelocity began working on Clarion# back in the days of .NET 1.1. Eventually they had to port to .NET 2.0, which was a major effort as there were a lot of breaking changes in .NET 2.0. As of 2.0, the core of the .NET platform is stable, and subsequent releases, 3.0, 3.5, and the forthcoming 4.0 (now in beta) all build on the foundation of the previous release. And with each release Microsoft typically adds some important new technology that developers, for some bizarre reason, expect to be able to use. And so SoftVelocity adds support for generics and lambdas and LINQ and all sorts of things that just don't exist in Clarion for Windows.

It's a brand new game of catch-up.

The role of Clarion#

All of this assumes that Clarion#'s rightful role among Clarion developers is as a primary .NET development language. If Clarion# *is* a primary language, then chances are you want it to have all of the functionality of VB.NET or C#, which means not lagging behind Microsoft terribly.

But if it takes the same number of lines of code to write something in Clarion#, and really the same code entirely, just with a slightly different syntax, then is it really worthwhile expending all that time and money putting the latest functionality into Clarion#? Mightn't it make more sense to see Clarion# as a way to move already-portable code to .NET, and just get on with the business of doing actual application development in VB.NET or C#?

It's a question worth considering, particularly because the pace of change out there doesn't seem to be slackening at all. If anything it's accelerating, as evidenced by the growing adoption of Windows Presentation Foundation, or WPF, as the new way to build user interfaces. I'll talk more about WPF, its implications for application architecture and the future of Clarion, in [Part 2](#).

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

Reader Comments

Posted on Tuesday, January 12, 2010 by Stephen Bottomley

Nice article Dave.

I'm not all that up on C# but I would think a couple of other reasons for sticking with Clarion# would be comfort or familiarity in areas like not worrying about type casting and case sensitivity (asuming these things carried over from C). Also those semicolons and braces especially add to coding nightmare of many C lookalike languages and is one of the big reasons I like Clarion (# or not).

To be honest I was also secretly pleased when SV announced they would go back to supporting the full stop (period) in Clarion# as a replacement for END even though I'd started replacing them in old code as I came across them :)

It's sometimes the little things.

Steve B.

Posted on Tuesday, January 12, 2010 by Dave Harms

Steve, personal preference and familiarity are definitely two points in favor of Clarion#.

Dave

Posted on Wednesday, January 13, 2010 by Michael Goosen

Looking forward to the next article!

The thing I love most about Clarion is that it gets working software out the door orders of magnitude faster than other dev environments (I also develop in Delphi and C#).

While I think it's important to keep abreast of the latest MS technologies, we have to maintain that [amazing] edge in development speed that every Clarion developer enjoys. I'm hoping we never lose that, and that Clarion# doesn't end up as a Visual Studio Emulator.

Posted on Friday, January 15, 2010 by Dave Harms

Michael,

Absolutely, the "instant application architecture" provided by the dictionary editor, the templates and the AppGen is what makes working in Clarion so productive. I really don't think there's much point to any version of Clarion for .NET that lacks the AppGen, and I'm sure SV realizes that as well.

Dave

[Add a comment](#)

Clarion Magazine

Sneak Preview: The New ClarionMag Site

by Dave Harms

Published 2010-01-14

As Clarion Magazine's publisher, at the beginning of each year I've looked forward to the coming twelve months with some anticipation of new Clarion product. That anticipation is still there, but at the beginning of ClarionMag's twelfth year I'm particularly excited about a new ClarionMag web site, currently under development.

As I've mentioned elsewhere in the mag, the current iteration is a combination of a custom Java web application, a MySQL database and a Clarion database admin program. This configuration has worked fairly well for the better part of ten years now, but it's getting a bit long in the tooth.

I started toying with the idea of a .NET rewrite of the ClarionMag web application around the time Clarion# was released. Although I've had a long history with Java, I wanted to take advantage of some of the work being done for web applications on the .NET platform.

There's still a lot of work to be done, but much of the core functionality of the ClarionMag server has now been ported to .NET. I don't yet have a beta site up, but I've been doing a lot of internal testing. In this article I'll show some screen shots of the new site and talk a bit about the underlying technology and some of the upcoming new features.

The home page

The overall look and feel of the ClarionMag home page hasn't changed that dramatically. Discreet areas on the home page still have the familiar rounded box with header appearance (Figure 1). The dark blue background is now only on the header and footer, not the entire page.

Some features are still missing from this early version of the new home page, including the survey. You can also expect to see more graphical elements to break up the text.

The screenshot shows the Clarion Magazine website home page. At the top, there is a dark blue header with the Clarion Magazine logo and the tagline "READ. LEARN. SOLVE." on the left, and a "Log On" button on the right. Below the header is a navigation menu with links for Home, Subscribe, E-Books, Print Books, News, Blog, Store, My ClarionMag, and Contact.

The main content area is divided into several sections:

- ClarionMag beta site news!**: A yellow-bordered box containing a welcome message to the beta release of Clarion Magazine 2010! dated 06/01/2009 7:58:59 AM. It lists new features already implemented:
 - An updated look (preliminary, subject to change)
 - Better home page navigation
 - A randomly generated "From the archives" article selection
 It also lists other new features in the works:
 - HTML comments
 - Post and vote on your Clarion 7.x showstoppers
 - Track your favorite articles
 - Rate third party products
 - and much more!
 A note says "Watch this space for further updates." and a pagination link "First Previous Next Last (page 1 of 2)".
- From the archives**: A white-bordered box with the article "Generating A Unique Registration Code" dated 26/01/2004 12:00:00 AM. The text describes a registration scheme using a product type code and a user name.
- Links**: A white-bordered box with the text "Coming soon!".
- Articles**: A white-bordered box with a pagination link "First Previous Next Last (page 1 of 174)". It features three article snippets:
 - Source Code Library 2009.11.30 Available** (07/12/2009 12:00:00 AM): The Clarion Magazine Source Code Library has been updated to include the latest source. Source code subscribers can download the November 2009 update from the My ClarionMag page.
 - ClarionMag on ClarionLive! Everything you ever wanted to know about Clarion Magazine and more...** (02/12/2009 12:00:00 AM): A Friday (Dec 4) webinar with Dave Harms, Clarion Magazine's editor and publisher, covering search engine tips, RSS feeds, and a site rewrite.
 - PDF for November 2009** (30/11/2009 12:00:00 AM): All articles for November 2009 in PDF format.
 - ClarionFAQ Update** (30/11/2009 12:00:00 AM): Here's what's happening at Clarion Magazine's new free-access Frequently Asked Questions.
- Clarion news**: A white-bordered box with a pagination link "First Previous Next Last (page 1 of 203)". It lists various news items:
 - DeckingPane Wrapper 1.07
 - Twitter Remote Control With C#/.NetTalk
 - EasyNavBar 1.00
 - Ingasoftplus Time Limited 25.00% Discount
 - CHT Web Client Server Videos
 - CHT Build 13D1.00
 - SV Blog: Clarion 7.1 Window Previewer
 - SV Blog: 32 bit ODBC in the 64-bit Operating System
 - Noyantis Support Ticket System
 - Noyantis User Forum
 - RPM C7 Cosmetic Oversight
 - EasyListView 1.02
 - ClarionLive! Double Header Week
 - Search Engine Profile Exchange November 03 2009 Release
 - EasyListView Version 1.02 Coming Soon, New Demo.
 - Noyantis Codejock Wrapper Templates Now Codejock v13.2.1

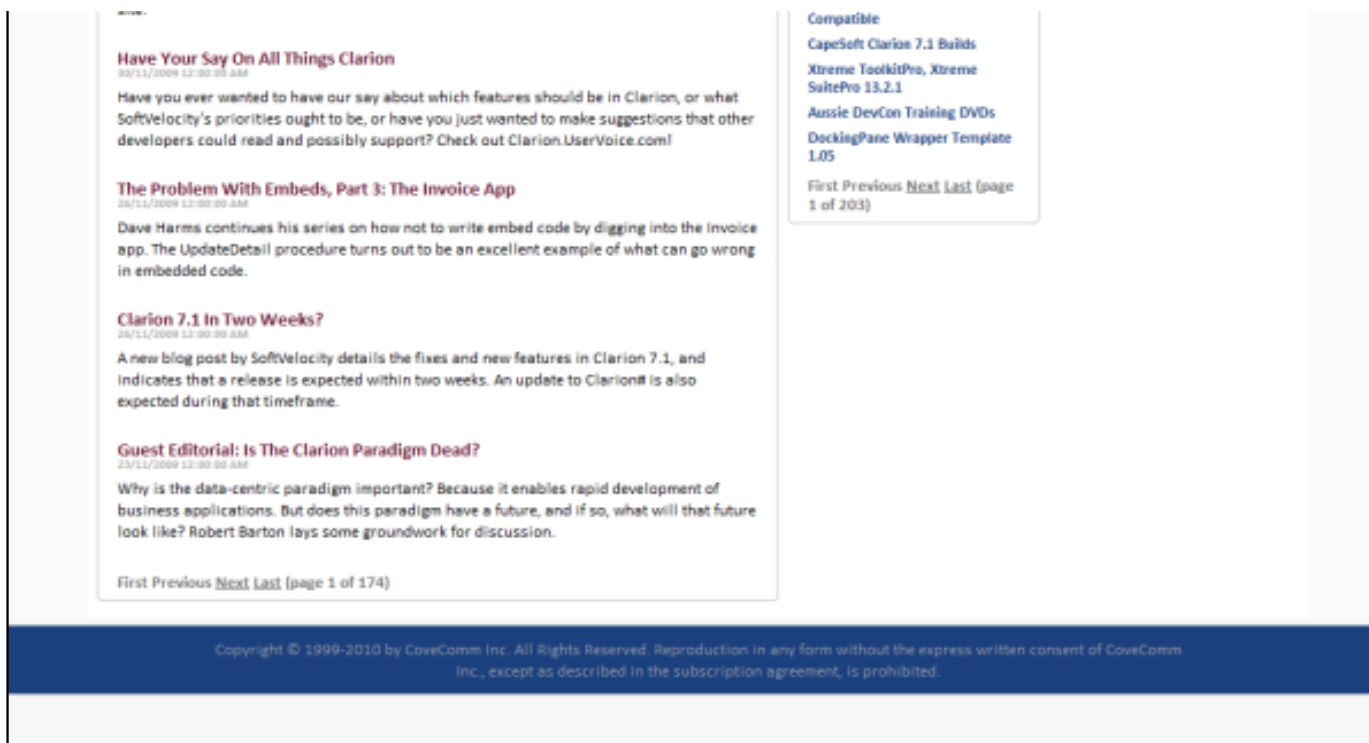


Figure 1. The preliminary design for the new home page (view full size image)

The menu bar has been redesigned (Figure 2) and is now more closely integrated with the header area.



Figure 2. The menu bar (with menu item showing hover highlight)

The content areas still have the rounded corners and the familiar header. Actually the rounded corners are done quite differently now. In the current site the rounding is all images; the new site uses CSS. Unfortunately (and perhaps predictably), IE is the only major browser that doesn't yet support rounded corners, although that's coming in IE9. As a result I'll probably set up the style sheet with an image to round just the upper left hand corner, and leave the rest square for IE users.

We've added navigation links to the content links on the home page. Currently these are text links; they may be changed to images in the release. This means you can easily page forward and backward through articles and news items, and your current choices are maintained for the duration of the session. If for instance you're on page five of the articles list, and you go to an article, when you come back to the home page you're still on page five of the articles list. Navigation controls are at the top and bottom of the lists.

On a technical note, although we're starting to use a bit more JavaScript in the site (mainly via JQuery) we've made every effort to keep things fully functional even if you don't have JavaScript enabled. You will however get an optimal experience with JavaScript turned on.

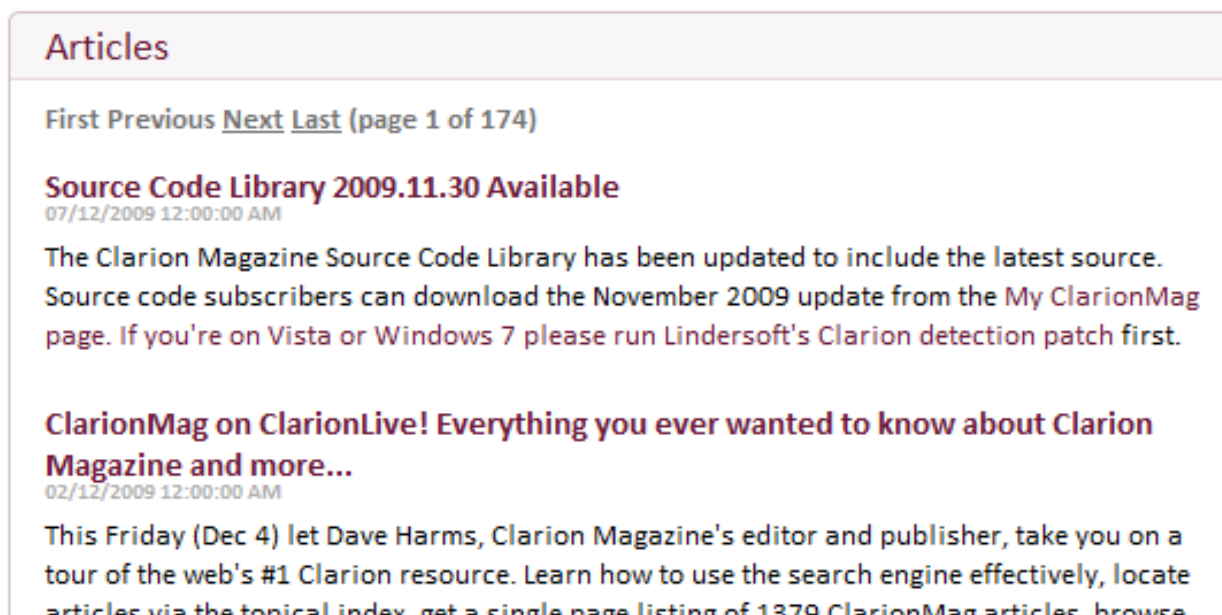


Figure 3 The articles list with navigation links

Random goodies

There's a lot of gold buried in the back issues. To help bring those articles to light we've added a "random article" feature (Figure 4). When you first visit the ClarionMag site a randomly selected article is displayed; you can manually refresh to get a new random choice.

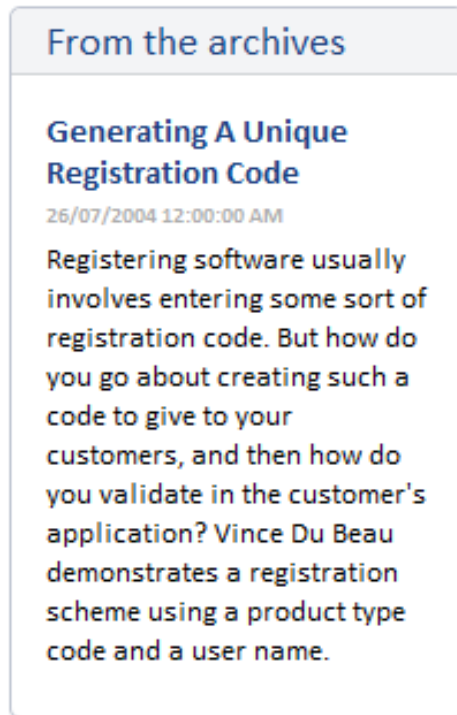


Figure 4. Random articles from the archives

Form based authentication

The current site uses Basic Authentication, in which the browser pops up the login window. The new site uses the more common Forms Authentication, as in Figure 5. There is a "Remember me" option.

Log On

Please enter your username and password. [Register](#) if you don't have an account.

Account Information

Username:

Password:

Remember me?

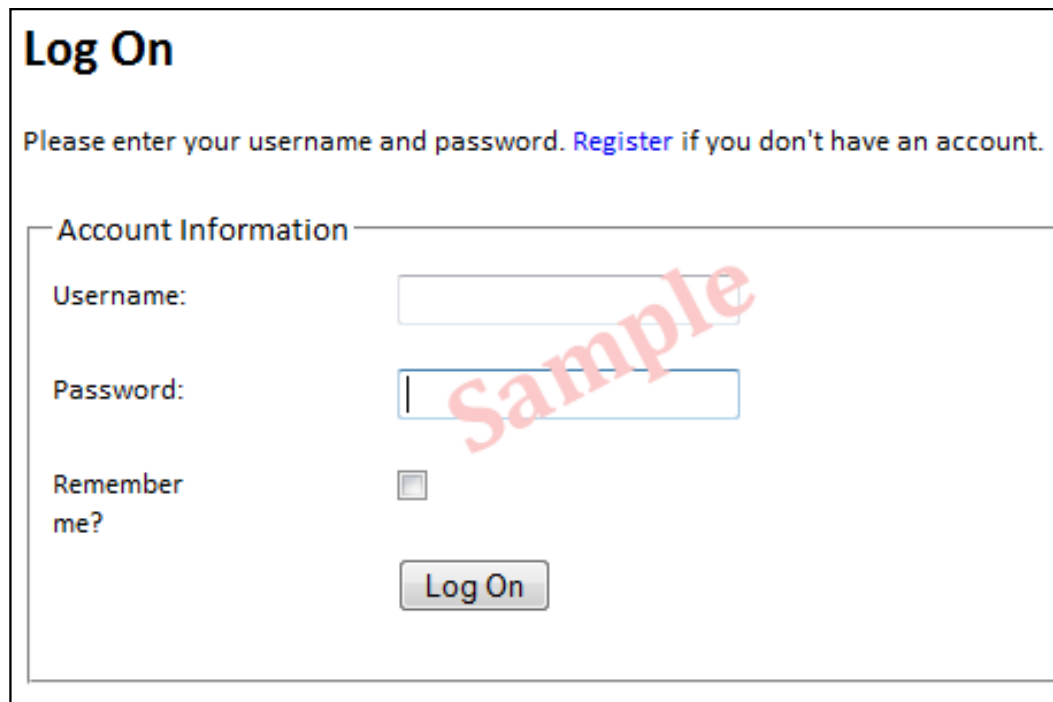


Figure 5. The new logon form

HTML comments

Commenting is not yet enabled on the new site, but we're looking at various options for allowing users to type their responses in HTML. Figure 6 shows a portion of an admin form for updating the "Beta news" section on the home page. This is a text control turned into an HTML editor using [jwysiwyg](#) and one tiny bit of JavaScript (the final choice for a JavaScript HTML editor has not yet been made).

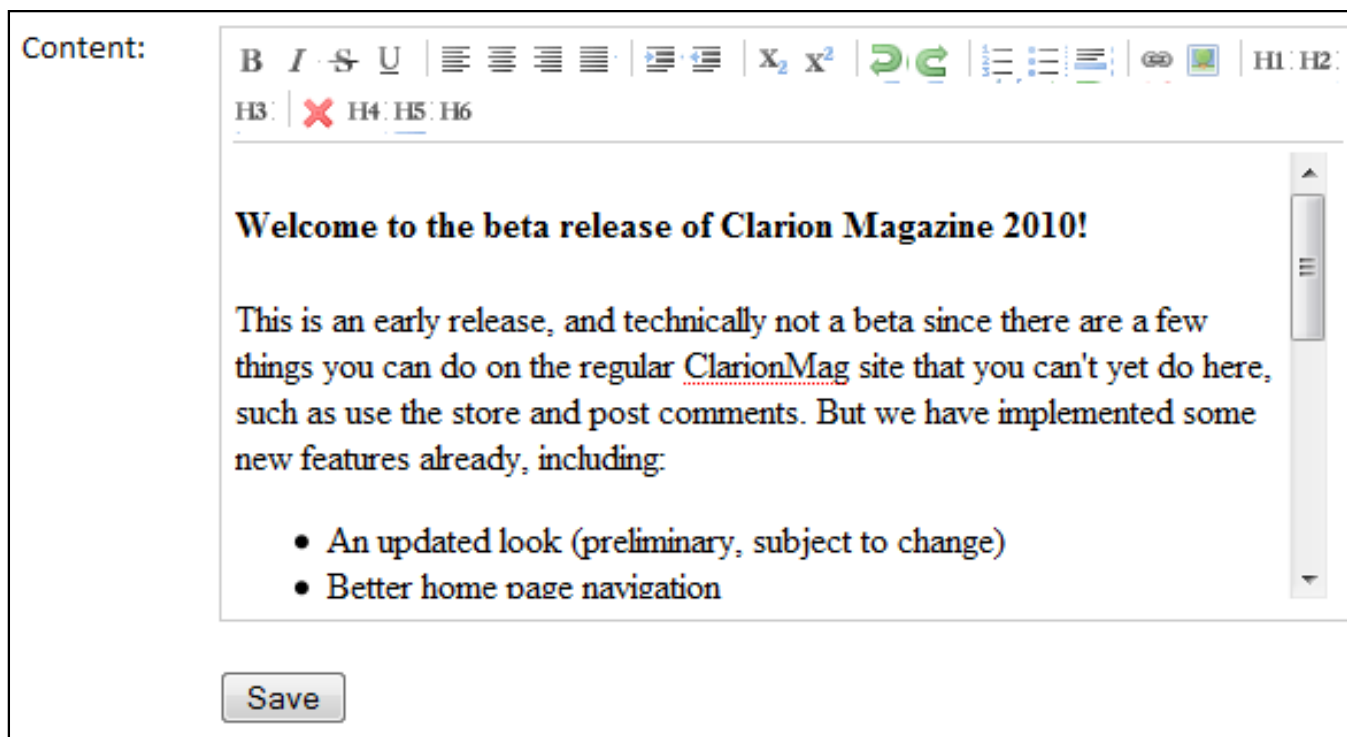


Figure 6. Editing an HTML post

Still to come

We're still adding some existing functionality besides comments (the store and the search engine are both only partially implemented as yet), but we've also got some new functionality in the works.

- Favorite articles. This feature has been on the wish list for a long time - a way to keep a personal list of your favorite articles for quick reference. Also in the works, an option to share your list with other subscribers.
- Better information about third party products. A long time ago ClarionMag had a third party tool directory, but it was difficult to maintain. We think we've come up with a better solution, and one that makes it easy for you to vote for your favorite tools.
- Post and vote on Clarion 7.x showstoppers. This one's a little less concrete right now, but we're going to take a stab at it.

We see this as a way to get a snapshot of the current state of major Clarion issues, so you can more easily decide if it's time

to upgrade. If this is successful we'll roll it out for other Clarion releases as well.

There's more on the way, but that's what you can expect to see first.

The technology

I mentioned that the current web server is written in Java. I used an MVC (Model-View-Controller) design and was quite happy with the results, so when it came time to port to Windows I spent some time looking around for MVC frameworks for .NET web apps. I began with Clarion# and the Castle Monorail framework, which worked well. But Microsoft has done a terrific job with ASP.NET MVC, improving on the work done in Monorail, so eventually I switched. Unfortunately I ran into some issues with ASP.NET MVC and the beta Clarion# compiler, so now all work on the new system is done in C#. That was disappointing, since I'd really hoped to use Clarion#, but not unexpected given that the latest ASP.NET MVC code often relies on the latest features in .NET.

On the plus side, Visual Studio is a pleasure to use. And Clarion isn't out of the picture at all. Web development, like almost any other kind of development, involves a certain amount of repetitious code, so I've also invested a fair bit of time in writing a set of templates for C7. Those templates generate C# code for use in Visual Studio. C7, by the way, is an excellent environment for writing templates, far superior to C6.

When will it go live?

There will be a limited private beta before the public beta. The ClarionMag office will be closed from January 23 to February 6, 2009 for winter vacation, if we don't get the private beta out before then it should happen shortly after the office reopens. Since no one outside of ClarionMag has played with the new site yet it's hard to say how long the private beta will last, but hopefully it'll be a quick process.

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

Reader Comments

[Add a comment](#)

Clarion Magazine

Keeping Separate Clarion Development Environments

by Benjamin Dell

Published 2010-01-18

As a contract software developer, I often run into situations where I have different projects from different clients that make use of the same templates or third party tools, but different *versions* of these tools or templates.

I also run into situations where I have to share a development environment with other developers and/or clients.

One way to handle this is to create multiple versions of the Clarion environment and then use a tool such as Lee White's [Clarion Switcher](#) to load the version of Clarion depending appropriate to the particular Project.

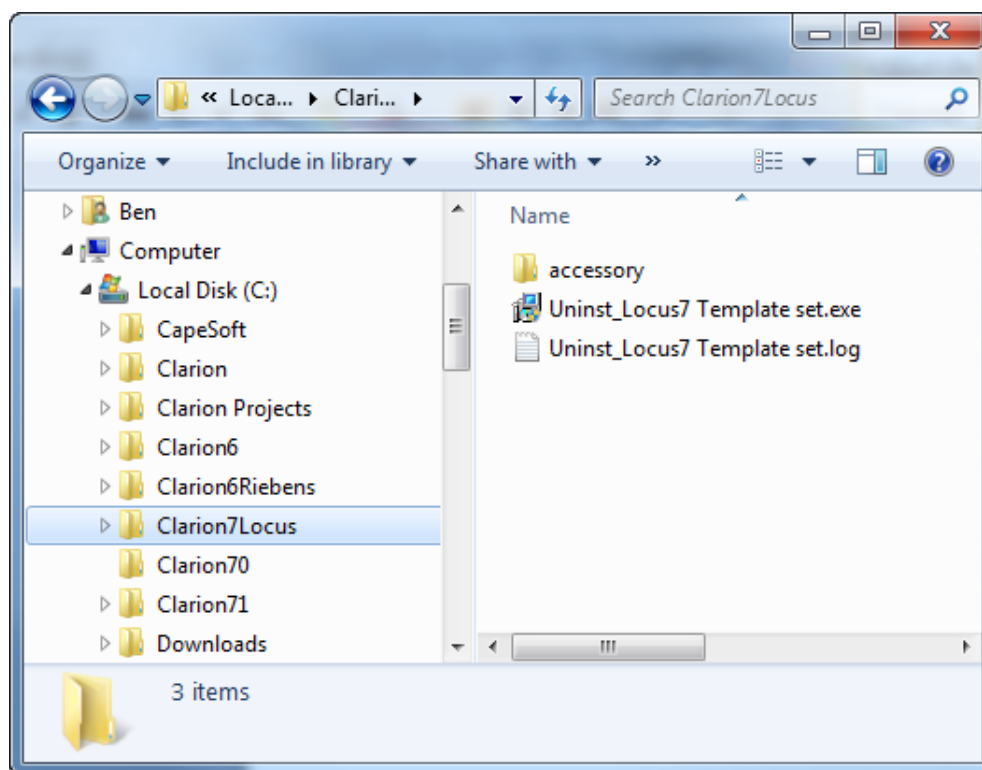


Figure 1. Example list of installed Clarion versions

What I will describe here is a different solution using the Redirection file that allows me to achieve the same end result with an added bonus of being able to easily save the whole development environment with the project, allowing me to replicate my dev environment per project without affecting other projects.

This will ensure that I never inadvertently update templates and then suddenly find that a project should not have updated, or implement a template in a project that the client does not have. It does, however, assume that I still keep different versions of Clarion installed, as needed; what it lets me do is keep multiple configurations for each installed version of Clarion.

I will be using Clarion 7.1 to demonstrate how easy this is.

The goal

From a contracting programmer's viewpoint, my goals are:

- To be able to work with different versions of the same templates and tools on different projects (solutions in C7),
- To be able to backup and restore the whole project and its development environment without influencing any of my other projects, and
- At the end of the project lifetime, I want to be able to distribute the source code and the development environment to the client (appropriate licensing conditions having been met) easily without any custom code for other projects included.

Having a look at Clarion's help file, I find the following paragraph:

Clarion's development environment sets the working directory to the one in which the current application or project file resides. Additionally, Clarion uses the redirection file (%ClarionRoot%\BIN\Clarion7.RED) to keep track of directories for the various application or project components. This redirection file tells the development environment where to find files and where to create new files.

But the most important part is the following:

The redirection file directory can contain the predefined %ROOT% macro. By default, the %ROOT% macro expands to the drive and path one level above that from which the environment program is executing. For example, if the environment program is in C:\C70\BIN, the environment substitutes C:\C70 for %ROOT%. The default redirection file uses the %ROOT% macro to work with Clarion's default directory structure, regardless of where you install Clarion.

You may override the %ROOT% macro's default substitution value by explicitly changing a value in the Redirection File options.

By changing the %ROOT% macro I can switch between different sets of supporting files, including templates, source code, and third party libraries.

Another very important aspect of Redirection files is that the local redirection file of the active directory is read first, and only if it does not exist is the redirection file in the Clarion Bin directory read.

Associating the dev environment with a project

First of all, I want to associate a specific version of my development environment with a specific project. Here are the steps to do that.

Create the Project directory. In this example I will create a Test Directory.

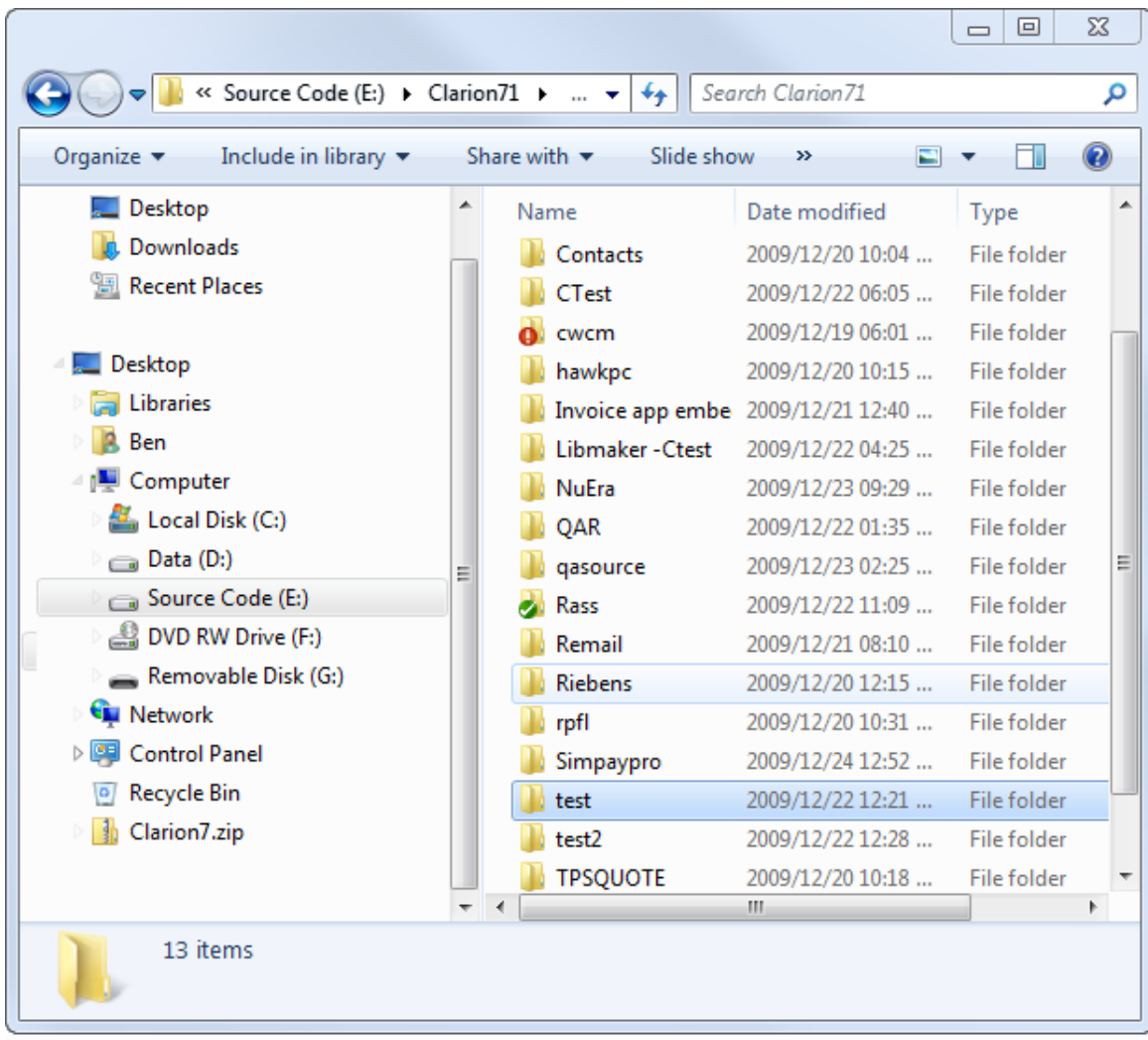


Figure 2. Blank Directory where my project will live

Copy the CLARION70.RED file into the newly created directory from the Clarion7\bin directory.

My normal Clarion redirect file looks like this (line breaks added):

```
-- Default Redirection for Clarion 7.1
```

```
[Copy]
```

```
-- Directories only used when copying dlls
```

```
*.dll = %BIN%;%ROOT%\Accessory\bin
```

```
[Debug]
```

```
*.obj = obj\debug
```

```
*.res = obj\debug
```

```
*.rsc = obj\debug
```

```
*.lib = obj\debug
```

```
*.FileList.xml = obj\debug
```

```
*.map = map\debug
```

[Release]

```
*.obj = obj\release
*.res = obj\release
*.rsc = obj\release
*.lib = obj\release
*.FileList.xml = obj\release
*.map = map\release
```

[Common]

```
*.tp? = %ROOT%\template\win; %ROOT%\Accessory\template\win; ↵
%ROOT%\template\Ctest
*.trf = %ROOT%\template\win; %ROOT%\Accessory\template\win; ↵
%ROOT%\template\Ctest
*.txs = %ROOT%\template\win; %ROOT%\Accessory\template\win; ↵
%ROOT%\template\Ctest
*.stt = %ROOT%\template\win; %ROOT%\Accessory\template\win; ↵
%ROOT%\template\Ctest
*. * = .; %ROOT%\libsrc\win; %ROOT%\libsrc\Ctest;%ROOT%\images; ↵
%ROOT%\template\win; %ROOT%\Accessory\template\win; ↵
%ROOT%\template\Ctest;.Images; %ROOT%\Images
*.lib = %ROOT%\lib
*.obj = %ROOT%\lib
*.res = %ROOT%\lib
*.hlp = %BIN%;%ROOT%\Accessory\bin
*.dll = %BIN%;%ROOT%\Accessory\bin
*.exe = %BIN%;%ROOT%\Accessory\bin
*.tp? = %ROOT%\Accessory\template\win; %ROOT%\Accessory\template\win; ↵
%ROOT%\template\Ctest
*.txs = %ROOT%\Accessory\template\win; %ROOT%\Accessory\template\win; ↵
%ROOT%\template\Ctest
*.stt = %ROOT%\Accessory\template\win; %ROOT%\Accessory\template\win; ↵
%ROOT%\template\Ctest
*.lib = %ROOT%\Accessory\lib
*.obj = %ROOT%\Accessory\lib
*.res = %ROOT%\Accessory\lib
*.dll = %ROOT%\Accessory\bin
*. * = %ROOT%\Accessory\images; %ROOT%\Accessory\resources; ↵
%ROOT%\Accessory\libsrc\win; %ROOT%\Accessory\template\win; ↵
%ROOT%\template\Ctest
```

Notice that many of these directories are under the directory specified by the Clarion %ROOT% Macro.

In Clarion 7, you specify the value of the %Root% symbol in Tools | Options | Clarion | Clarion for Windows | Versions (Figure 3).

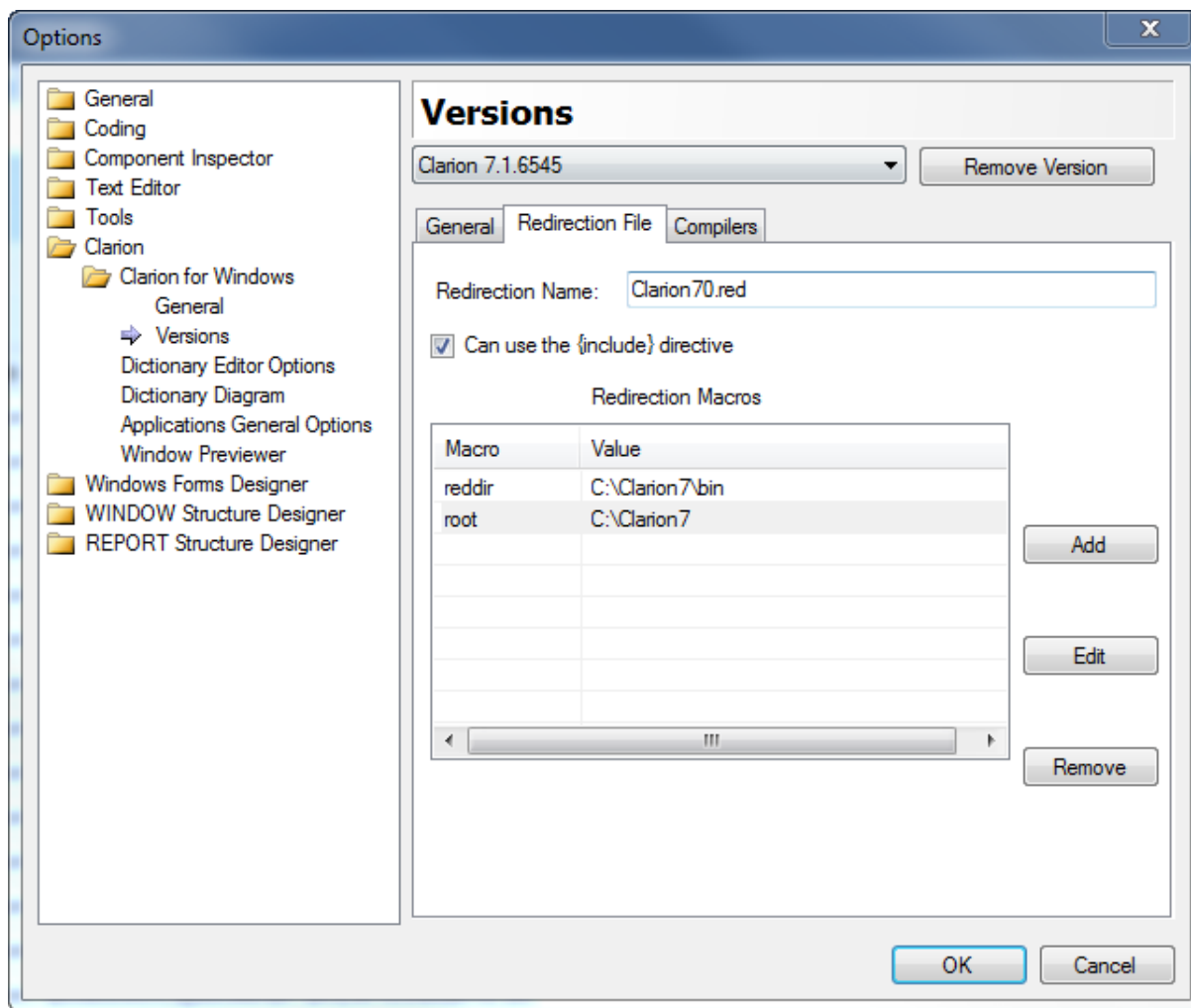


Figure 3. Standard Root Macro

I want to achieve two things with this particular example:

1. Combine all my templates used in a project into the same directory and
2. Make my TEMPLATE and LIBSRC and LIB directories unique to the current project.

For that I create a unique %ROOT% replacement for each project. In this instance I will create a %TESTROOT% macro as this is the TEST project.

Here is my edited of the Redirection file (using Notepad++).

```
-- Default Redirection for Clarion 7.1 Project Test
```

```
[Copy]
```

```
-- Directories only used when copying dlls
```

```
*.dll = %BIN%;%TESTROOT%\bin
```

```
[Debug]
```

```
*.obj = obj\debug
```

```
*.res = obj\debug
```

```
*.rsc = obj\debug
```

```
*.lib = Make
```

```
*.FileList.xml = obj\debug
*.map = Make
*.clw = Make
*.inc = Make
*.exp = Make
```

[Release]

```
*.obj = obj\release
*.res = obj\release
*.rsc = obj\release
*.lib = Make\lib
*.FileList.xml = obj\release
*.map = Make\map
*.clw = Make\clw
*.inc = Make\inc
*.exp = Make\exp
```

[Common]

```
*.tp? = %TESTROOT%\template\win
*.trf = %TESTROOT%\template\win
*.txs = %TESTROOT%\template\win
*.stt = %TESTROOT%\template\win
*.gif = .\Images;%TESTROOT%\images;
*.ico = .\images;%TESTROOT%\images;
*.JPG = .\images;%TESTROOT%\images;
*.cur = .\Images;%TESTROOT%\images;
*.bmp = .\Images;%TESTROOT%\images;
*.* = .; %TESTROOT%\libsrc\win;%TESTROOT%\template\win
*.lib = %TESTROOT%\lib
*.obj = %TESTROOT%\lib
*.res = %TESTROOT%\lib
*.hlp = %BIN%;%TESTROOT%\bin
*.dll = %BIN%;%TESTROOT%\bin
*.exe = %BIN%;%TESTROOT%\bin
```

As you will notice, I dump all my templates into one folder, all my class files in one folder, all my lib files into one folder, etc.

Now I create all the folders under my TEST project directory that I will need and copy all my files used from my standard Clarion installation into the appropriate folders underneath the project directory.

How I copy and update these files (as when Clarion releases a patch) is outside the scope of this article, but I have a mechanism in place to update templates and tools that needs to be updated on a project by project basis. There are lots of directory compare utilities out there that can help with this job.

This gives me a typical project directory structure as depicted below.

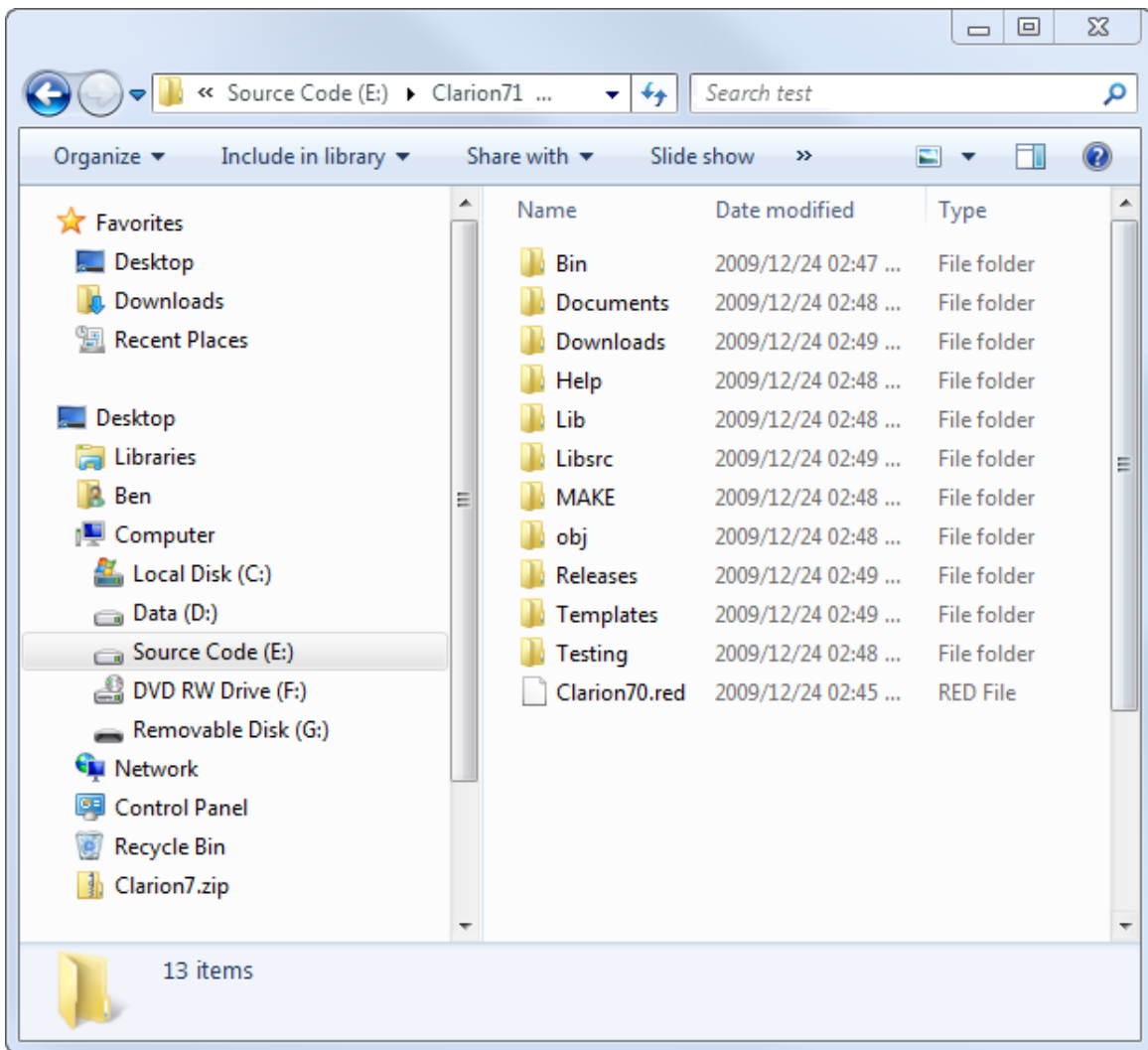


Figure 4. Project directory structure.

This technique allows me to put the TEST root directory under version control and also if need be I can re-distribute the whole project with all the associated development environment.

The last thing I have to do is tell Clarion what the %TESTROOT% macro is that I defined earlier in my new project based Redirection file. In Clarion 7 I simply add a new redirection macro.

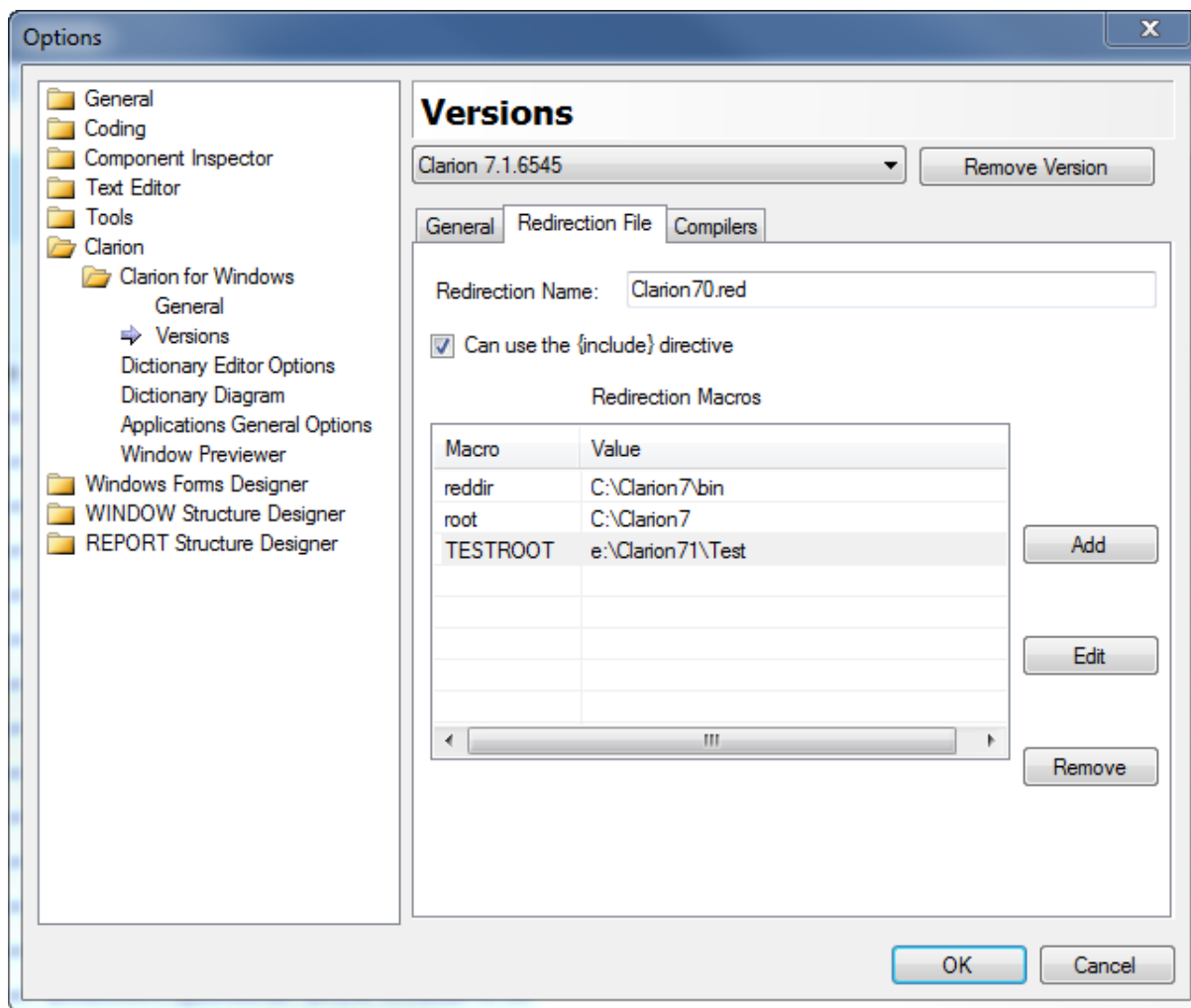


Figure 5. New Macro defined in Clarion IDE

Now whenever I select my TEST project from the IDE, the template, lib and libsrc directories used by my project are those defined in the Redirection file for the TEST project and not the normal ones as default installed by Clarion.

In one go, I have version controlled not only my project, but also the unique development environment for that project. This makes for very effective control in Contract development situations and also in multi developer situations on the same project.

NOTE: If you use this technique, make sure that the other developers on the project have a valid license to use any templates or tools included in the project.

Keep in mind that this technique is *not* a cross-version technique; if your Clarion version is, for instance, Clarion 6.2, you cannot restore the development environment back to a Clarion 6.3 version. You have to keep separate installs of each version of Clarion. That's easy to do with C7; just install each version into its own folder. You do the same with C6, but versions of Clarion prior to C7 use a standard configuration file and a PATH setting. In the case of C6 and earlier versions you can take the redirection file approach for all the supporting code and use Lee White's ClarionSwitcher to manage the Clarion IDE executables.

Directory structure

I store all my projects and solutions on a separate drive on my computer and on the root of that drive I have a folder for each version of Clarion. In my case, I have two subfolders under for instance E:\Clarion70 named Riebens (for internal projects) and Clients (for client projects). Each project or solution then has their own separate folders underneath

each of these two main folders (Figure 6).

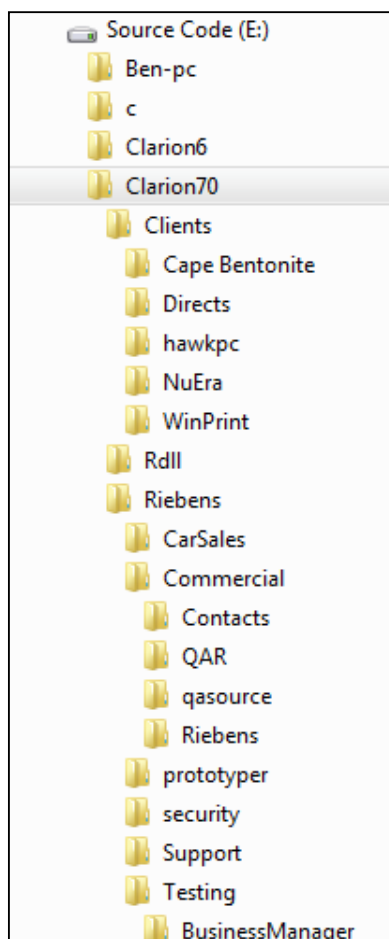


Figure 6. The directory structure

This allows me to visually sort my projects according to the Clarion version with which they are developed.

Summary

Customizing the %ROOT% macro makes it easy to maintain complete Clarion configurations on a per-project basis, particularly when using Clarion 7.x. It's a little more work in pre-C7 versions because the Clarion IDE has some file and path dependencies, but it's still pretty straightforward.

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

Reader Comments

[Add a comment](#)

Clarion Magazine

ClarionMag Private Beta Update, And Some SQL

by Dave Harms

Published 2010-01-21

The new Clarion Magazine web site has gone into private beta. Technically it's more alpha than beta, since all the functionality of the current has not yet been ported. The look and feel has been fine tuned (still, well, *always* subject to change) and a couple of bugs have been fixed.

The private beta is on hold for a couple of weeks while the office is closed for vacation, but as soon as we're back we'll be working toward the public beta.

I've included another screen shot of the new site, now running on IIS.

Deploying a development web site to a beta production server can be a painful process, but this was one of the smoothest implementations I've experienced. Setting up the site on IIS7 (which I'd previous installed) was a breeze. All I really had to do, other than add the new site entry in the ISS7 Manager, was install the PostgreSQL development database, copy across the needed DLLs and supporting files for the web site itself and change a few config settings (none of which were related directly to IIS). The whole process took about two hours, a good chunk of which was copying files to the new server across a not particularly fast uplink

I've heard lots of horror stories about installing web apps on earlier versions of IIS. With IIS7 and ASP.NET MVC, it's little more than an XCOPY.

Among other things we've added main menu navigation highlighting. In Figure 1 I'm hovering the mouse over the News tab. This is a Javascript effect; if you don't have Javascript enabled then you'll just see the menu text highlighted in white, not the entire tab.

The screenshot shows the Clarion Magazine website interface. At the top is a dark blue header with the logo and the text "CLARION MAGAZINE READ. LEARN. SOLVE." and a "[Log On]" link. Below the header is a navigation menu with tabs for Home, Subscribe, E-Books, Print Books, News, Blog, Store, My ClarionMag, and Contact. The "News" tab is highlighted with a white background and a mouse cursor is hovering over it. The main content area features a yellow banner with the text "ClarionMag beta site news!" and "Welcome to the beta release of Clarion Magazine 2010!". Below this is a list of new features and a "From the archives" section with a link to "Receiving Email With MAPI (Part 2)".

Watch this space for further updates.

[First](#) [Previous](#) [Next](#) [Last](#) (page 2 of 2)

Articles

[First](#) [Previous](#) [Next](#) [Last](#) (page 4 of 174)

Aussie DevCon Day 1 Notes
10/12/2009 12:00:00 AM
The first training day of the Aussie DevCon is in the books, with Bob Foreman providing training Bruce Johnson showing a preview of NetTalk 5, and ClarionLive! providing a video link.

Source Code Library 2009.09.30 Available
10/5/2009 12:00:00 AM
The Clarion Magazine Source Code Library has been updated to include the latest source. Source code subscribers can download the September 2009 update from the My ClarionMag page. If you're on Vista or Windows 7 please run Lindersoft's Clarion detection patch first.

PDF for September 2009
9/30/2009 12:00:00 AM
All articles for September 2009 in PDF format.

AES Encryption And Test Vectors
9/29/2009 12:00:00 AM
AES encryption is easy to implement in Clarion, thanks to Carl Barnes' example from a few years back. But how do you ensure the encryption is working correctly? With test vectors, of course.

The Problem With Embeds, Part 1: What Went Wrong?
9/28/2009 12:00:00 AM
As Clarion developers we (well, most of us) live and die by embed code. Embeds are, after all, at the heart of Clarion's effectiveness. They're the primary way we add custom code to applications that are otherwise template-generated. And if you're like most Clarion developers, you're probably using embed points the wrong way. Without realizing it you're making your applications more difficult to maintain, harder to debug, and almost impossible to document.

Making TPS Superfiles
9/28/2009 12:00:00 AM
In previous articles Steve Parker explained what TPS superfiles are and how to name them. In this installment Steve shows how to create superfiles from existing TPS files.

Template Logging/Debugging Revisited
9/23/2009 12:00:00 AM
There really is no such thing as a template debugger, much as that would make life easier for many Clarion developers. But thanks to Mark Goldberg and Russ Eggen, there is a nifty technique for logging template output that makes template debugging much easier. Dave Harms revisits the code in C7 and explores some of the advantages of using DebugView to log messages, whether those messages come from templates or from regular source code.

Friday ClarionLive Webinar: SQL Super Panel
9/24/2009 12:00:00 AM
Clarion Magazine's Dave Harms will be moderating the first ever ClarionLive SQL Super Panel, featuring David Swindon, Jim Morgan, Joe Tailleir, and Shawn Mason. Topics include: Definition of SQL; The range of SQL approaches; The SQL perspective; The Clarion perspective; Advantages/Disadvantages as compared to TPS; Questions you need to ask yourself; and more. And stay tuned for a special announcement at the end of the webinar!

[First](#) [Previous](#) [Next](#) [Last](#) (page 4 of 174)

Clarion news

[First](#) [Previous](#) [Next](#) [Last](#) (page 2 of 203)

[SkinFramework Wrapper Template 2.00](#)

[ReportControl Wrapper Template 1.11](#)

[TaskPanel Wrapper Template 2.00](#)

[PropertyGrid Wrapper Template 1.10](#)

[CalendarPro Wrapper Template 2.00](#)

[CommandBars Wrapper Template 2.01](#)

[ShortcutBar Wrapper Template 1.21](#)

[RPM Update and Production Schedule](#)

[SetupBuilder 7.0 Build 3754](#)

[SetupCast 1.6.0](#)

[EasyListView 1.01](#)

[DHC 1.7.0.0 Two Year Anniversary](#)

[SetupBuilder Ready For Windows 7](#)

[EasyCOM2INC 2.10](#)

[900 Data Management Concepts & Terms](#)

[Clarion Handy Tools On ClarionLive!](#)

[ClarionMag's New Free FAQ Site](#)

[Aussie DevCon ClarionLive Report Time Change](#)

[IF Arnold Can Do It, So Can You!](#)

[Clarion Third Party Profile Exchange October 1 2009 Release](#)

[First](#) [Previous](#) [Next](#) [Last](#) (page 2 of 203)

Copyright © 1999-2010 by CoveComm Inc. All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the subscription agreement, is prohibited.

Figure 1. The beta home page as of January 21, 2010 (view full size image)

Users!

An odd problem cropped up shortly after I set up the private beta. One of my testers discovered he couldn't log in using the same username he'd always used on the current site.

It turned out he was lowercasing the first letter of his username, which was actually a capital letter.

Passwords have always been case sensitive in ClarionMag, but usernames are not. In part that's because the current site runs on MySQL, and when you retrieve a record by a string value MySQL does a case insensitive match. If the stored username is IAmAClarionMagSubscriber and the user types IamAClarionmagSubscriber, no problem, MySQL still pulls back the record and everything's fine.

But PostgreSQL does a case sensitive match by default. It also means that you have to do some extra work if you want a case insensitive unique constraint on a column in a table, as I do with usernames.

The hard way

The first thing I thought of was to create a new column with the lowercase value of the username, and update that column automatically with a trigger. Here's the SQL code to add the column:

```
ALTER TABLE users
  ADD COLUMN uniqueusername character varying(60);
```

Since this table already contains data, I need to do a one-time update of the column:

```
update users set uniqueusername = lower(username);
```

And then I need to add a unique constraint:

```
ALTER TABLE users ADD CONSTRAINT
  users_uniqueusername_uniqueconstrint
  UNIQUE (uniqueusername);
```

But that's not all. I want that column to be updated when new data is added or data is changed. So I need a stored procedure:

```
CREATE OR REPLACE FUNCTION user_set_uniqueusername()
  RETURNS trigger AS $user_set_uniqueusername$
  BEGIN
    new.uniqueusername = lower(new.username);
    RETURN NEW;
  END;
$user_set_uniqueusername$ LANGUAGE plpgsql;
```

And finally I need a trigger to call that stored procedure whenever a record is added or changed:

```
CREATE TRIGGER user_set_uniqueusername
  BEFORE INSERT OR UPDATE ON users
  FOR EACH ROW EXECUTE PROCEDURE user_set_uniqueusername();
```

Now I can search for my username on the uniqueusername column and everything's cool. But yikes! That's a lot of code to do something MySQL does by default.

The easy way

Happily, there is an easier way. PostgreSQL supports something called *functional indexes* where you create an index that uses

a function, such as (you guessed it) `lower()`.

Here's how I create a functional lower case index on my users table:

```
create unique index users_username_lowercase  
on users (lower(username));
```

Now I can use that index by adding the PostgreSQL `lower()` function to the query:

```
select * from users where lower(username) = 'iamaclarionmagsubscriber';
```

That works, and it's a whole lot less code! The only downside is I have to use `lower()` in my query - simply lowercasing the search term isn't enough.

I got blindsided by this issue because I hadn't done a good enough job of testing, so one of the first things I did was go back and write some new [unit tests](#). But no amount of testing is going to unearth all the issues; your users will always find things you overlooked.

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

Reader Comments

[Add a comment](#)

Clarion Magazine

Clarion 7.1 Build 6695 Release Notes

by Dave Harms

Published 2010-01-21

SoftVelocity has released 7.1.6695, with a number of important fixes. Unfortunately Clarion Magazine's office will be closed for winter vacation from January 23 - February 6, so we won't have a review up for a couple of weeks. That's not necessarily a bad thing, as in the past it's usually taken a few weeks for the users to shake out any critical bugs.

Meanwhile, here's a look at some of the highlights from this build:

- If you double-click on a file, key or field in the data pad while a text editor is open, that label will be pasted into the text editor.
- Better CSIDL support
- A new printer dialog that lets you invoke printer setup for an opened report.
- New menu themes and runtime support for modifying themes
- More efficient app recovery log writing, and a way to disable the recovery log
- Improvements to the IDE's command line interface
- Numerous bug fixes to the IDE and the AppGen
- Runtime library fixes
- Conversion (from C6) bug fixes
- Fixes to code generation errors
- A new `LARGE_ADDRESS` directive to let 32 bit apps use more memory when run on a 64 bit operating system.

You can read the [full list here](#).

I don't see a switch to disable owner drawn menus in this release, so those who want OS menus are still out of luck, although presumably the menu-related template changes offer some relief.

I can verify that the delays caused by writing to the recovery log have been greatly reduced, although you may find there are now delays in code generation while the .AP~ file is updated. On my machine disabling the recovery log removes these delays completely.

And Steve Parker reports his date fields are working correctly.

Feel free to post your impressions of C7 as comments to this article.

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

Reader Comments

[Add a comment](#)