Insights and Rants

News and general opinion, often privacy, security or computer related, but could be about anything really, including religion, politics, the environment, business or audio books. "Never ascribe to malice, that which can be explained by incompetence/stupidity/neglect." ~ Hanlon's Razor

Saturday, October 14, 2023

Clarion Fixer (Part 1): Using the App

One of my pet peeves with Clarion is that the applications it generates by default look like they were written for Windows 95, and for the current version of Windows. In modern versions of Windows, Microsoft has "fixed" the old look by making all the buttons and controls flat, but Clarion insists on using the "Microsoft Sans Serif" font that was introduced in 1997.

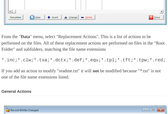
It looks odd, especially since I have already used and discarded Windows and Subsons, for much the same reason. I seven liked Arial (1982) it skipped with Windows 3.1 and has been severely over-used. "MS Sans Serif" is a bitmap font used in the 1992 version of Windows. I want my programs to look like they were written recently, not 30 years ago.

My first attempt at fixing the from in my application didn't work. I was trying to fix things in the wrong place. But I later discovered that I could use a file utility called Search and Replace to fix the Clarion styles and templates. So when I create the app, I don't have to waste time fixing up form all over the place. This got me thinking: can't I automate this? Clarion doesn't have an updated accessories for Clarion. I don't want to have to fix up the fonts yet again.

Introducing **dfFixer**, which is freeware. It's not signed, and some anti-virus engines on VirusTotal think it's a trojan, but I think they are getting false positives. Naturally, it is written with Clarion 11.1, and uses the CapeSoft StringTheory 2 (B57) and WinForms 1 ($100) accessories. If you have those, you can download and modify the code here. The .ne file doesn't contain any .exe or .dll files. The .cw file contains those. If not, you can read the code in the module files. Extract the .zip file into a suitable folder, if you wish.

The first thing you need to do is make a copy of your Clarion folder, so you can see what changes have been made, and how they work. I have copied mine to 'E:\Clarion11.1.0a'.

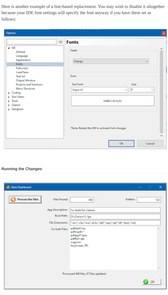
Exploring dfFixer



Run the **dfFixer** app, and from the main menu choose "**Data**" -> "Application Settings".

Click on the ellipse to the right of the 'Root Folder' field.

Select the backup folder and click 'OK'. Note there is a list of 'Exclude Files'. Some changes, such as the "Creator New" change, will cause trouble in the PDF related files. So you can decide whether a particular action must avoid modifying these files. Click 'OK'.



From the "**Data**" menu, select "Replacement Actions". This is a list of actions to be performed on the files. All of these replacement actions are performed on files in the "Root Folder" and subfolders, matching the file name extension.

`*.inc;*.clw;*.txa;*.dctx;*.def;*.equ;*.tpl;*.ff;*.tpw;*.red;`

If you add an action to modify "readme.txt" it will **not** be modified because "txt" is not one of the file name extensions listed.

General Actions

Here is the first and one of the most generic actions. It applies to all files, except a small number on the "Exclude Files" list.

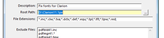
This is even more general because it applies to the excluded files as well as any other file that still mentions 'MS Sans Serif'.

Specific Actions

This action is specific to one file, and fixes up a form where the font names wrapped around over 5 lines.

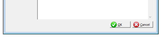
This action is specific to the TTT extension only. It fixes a general replacement from "Arial" to "Segoe UI" amended for forms, and changes the font to "Calibri" for a setting to do with reports. If you use a wild card file this is the file name, be sure to omit the '.' after the wild card, and use only one file name. The software isn't expecting multiple file names in this field.

Line-Based actions



Sometimes a search and replace across an entire file is not what we had in mind, and can cause havoc. Here is an example. The term 'FONTSETUP(vall' appears in multiple places, so I only want to change this specific one. Note that I have included the first part of the next line as well, to prevent the same replacement happening again if the file is scanned more than once.

Here is another example of a line-based replacement. You may wish to disable it altogether because your IDE font settings will specify the font anyway if you have them set as follows:

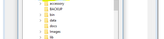


Running the Changes

From the "**Data**" menu, select "**Main Dashboard**" to view all the settings. Then click on "Process the Files" to read the file structure and make the changes. Most of the work is now being done by StringTheory and the process is pretty fast. When done, there is a file created 'dfFixer.log' in the root folder, which is automatically opened for you. This lists all the files affected by changes.

Checking the Changes

It's important to check the changes, especially if you have added in new ones. before dfFixer saves a changed file it checks to see if a **'before'** file exists. This is a file with the '.bak' added to the end. It's the original file that was renamed, and will still have the original date on it.



Here is a file comparison using the ExamDiff Pro utility, but there are others you may prefer. The original file is at the bottom, and the new file is at the top. It's important to do this because you don't want to screw up your actual working environment and have to reinstall things.

Once you apply the changes to your working environment, remember that none of your existing apps will have changed. Only **new** forms, reports and apps, and template-generated items will be affected by these changes.

Why not just pay CapeSoft $97 for their AutoFixer product? My main objection is that this product allows the user to make bad font choices, and select something horrible like "Comic Sans". Horror! Also, fonts have different widths and spacing, and I don't want my forms to look different if my customer happens to crying to fit in the space intended by Arial Narrow. Legibility is more important than looks.

In the next part, we will take a look at some of the programming techniques (good and bad) used in this application.

[Clarion Fixer (Part 1): Using the App]
[Clarion Fixer (Part 2): The Code]
[Change Mode vs View Mode]
[Clarion Build Events]
[Introducing Clarion FileManager (Part 1)]
[Introducing Clarion FileManager (Part 2)]
[**Clarion Resources** and Products]

Logo credit: Toolbox icon created by Rosenfeld-on-Premises - Flaticon.

Posted by Dom Edwards at 2:25 PM

Labels: CapeSoft, Clarion, dfFixer, Fonts, Productivity, StringTheory, WinEvents

No comments:

Post a Comment

Home Older Post

Subscribe to: Post Comments (Atom)

Copyright © 2006-2022 by Dom Edwards. Off-topic or spam comments will be deleted. Powered by Blogger.

Useful Links

Clarion Resources and Products

Installing Clarion 11 on Windows 10

Introduction to Clarion 11

Getting Started with Clarion (Part 1)

Getting Started with Clarion (Part 2)

Backing up your Clarion Projects

Learning Clarion (Part 1)

Learning Clarion (Part 2)

Learning Clarion (Part 3)

Learning Clarion (Part 4)

Learning Clarion (Part 5)

Learning Clarion (Part 6)

Learning Clarion (Part 7)

Adding Extensions to Clarion

Clarion FileManager (Part 1)

Clarion FileManager (Part 2)

Change Mode vs View Mode

Fact-Reviews - dxfrag.web.site still under construction

Jubilee Exams Art Studio

Jargon Free Guide to Computer Security

Lookup SA SMS numbers

Black and White Inc.

Lifetime Music Radio, also known as LM Radio

Contact Details

Dom Edwards
dom@dfuzzy.org.nz
mobile +1763-322-4966
Skype: dfuzzy.org.nz

Please do not contact me for spam and scam/phishing. Do NOT use my email address, phone number or Skype name for unsolicited marketing messages. Do not send me unsolicited marketing messages. You can follow me on Twitter @dominoedwards or on Instagram as dominoedwards