SizingServers.Util

Intro

This library holds a range of utils to make ones' life easier.
You can find the (64 bit, .Net 4) binaries, in the Build folder. The c# source code can be found in the Visual Studio 2015 solution.

Examples are available for most utils via the examples WinForms application.

Contents

BackgroundWorkQueue: You can enqueue delegates here to be processed by a background thread. Handy for work that needs to be done in order without blocking the main thread.

CompilerUnit: Facilitating compiling string input. Compiles C# .Net 4(.#).

Countdown: A countdown build using a timer.

DataGridViewImageCellBlank: A WinForms control. Draws a blank image for replacing the ugly, unneeded 'no image' image if no image is set. Put an instance of this in the CellTemplate of a DataGridViewImageColumn.

FastObjectCreator: To create an object using intermediate language instead of Activator (if possible). Only an empty constructor can be used. Used to be faster than Activator (no checks), but maybe not anymore. I have to do some testing.

FindAndReplace: Find and replace made easy (no complicated regex stuffs). Uses old-school Google tokens. A WinForms user control that encapsulates this functionality is also available.

FunctionOutputCache: Inspired by APC. Output that stays always the same, but takes long to calculate, gets cached. For example: A datatable. See FastObjectCreator code to see how it works.

InputDialog: A blast from the past, the VB6 input dialog in 'modern' WinForms.

KeyValuePairControl: WinForms control to visualize a key value pair.

RemoteDesktop: Enables using the default Windows RDP client from within your app.

StringUtil: String helper class. With this you can format numeric values and reverse strings.

SynchronizationContextWrapper: A wrapper for SynchronizationContext, so SynchronizationContext is available throughout the entire application. A SynchronizationContext is for synchronizing data to another thread, for instance to the main thread. Handy if you want to update the GUI, e.g. a System.Windows.Forms.Label, from another thread. It is the most reliable way to do something like this IMHO.

TabControlWithAdjustableBorders: You can choose which borders are visible or not for this WinForms control. This is usable for making a GUI cleaner when you have alot of nested controls.

Tracert: Like the cmd tool. A WinForms user control that encapsulates this functionality is available.