Howto Use VIDE/XIDE with XAML

Preface

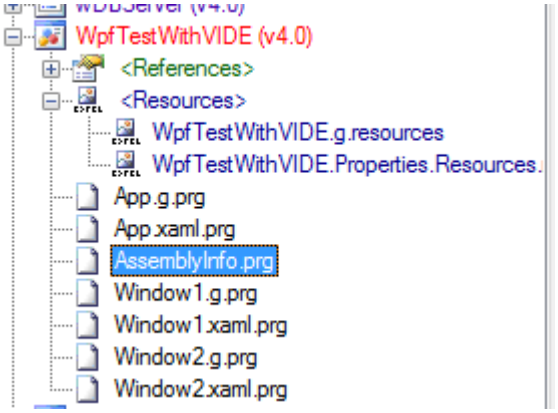
Unfortunately VIDE/XIDE has no support for XAML. I had contacted Chris Pyrgas, the author, various times about this, but unfortunately he was not able to add XAML support.

Personally I prefer VIDE/XIDE very much over Visual Studio – it is more code oriented, the indenting works a lot better, and ist user interface is cleaner. Generally I'm much more productive with XIDE/VIDE than with Visual Studio. And VIDE/XIDE has an AutoExport feature (I'm backing up to my network drive when in office, and to a SD card when out of office), the possibility to generate export files, and is much faster both in startup and work.

And since WPF is the preferred tecnology for GUI applications under the Microsoft Windows operating system, there was a need to find a possibility to combine development of XAML forms with VIDE/XIDE development.

First solution

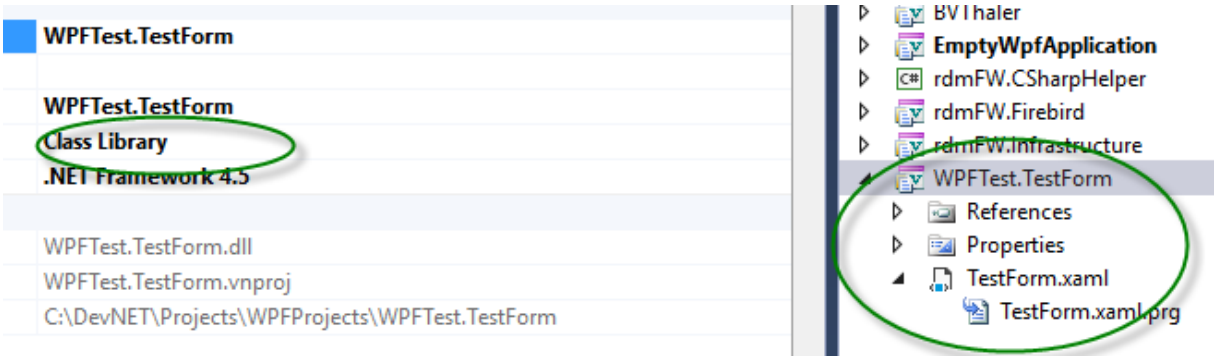
Given the fact that for XAML development Visual Studio (VS) has to be used, there was a first approach to include simply the VS generated files in the VIDE project as external files, the BAML as ressource, then the .g.prg file from the obj subdirectory and the .prg file for the class. This approach was found by Chris Pyrgas itself.



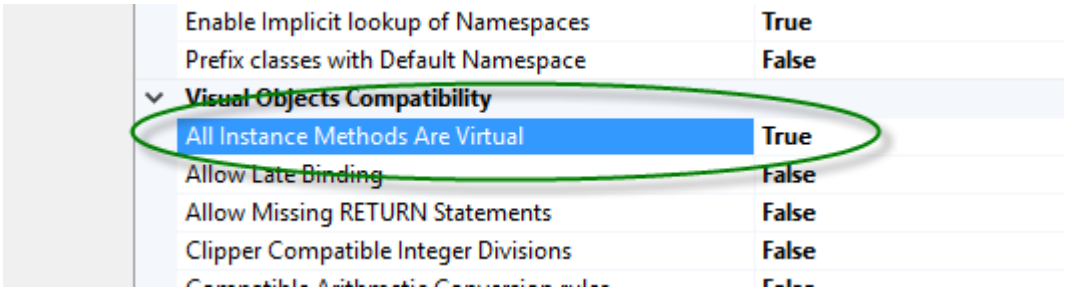
This works, but in my opinion it is too much work, I needed an easier option.

Final solution

Since with the .NET framework it is possible to mix languages, the next approach was to build the XAML window entirely in VS and include it as DLL in VIDE/XIDE:

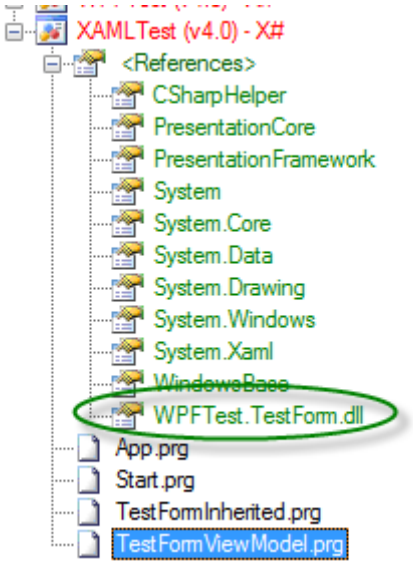


There is one compiler option that must be set to compile the form library:



Otherwise the library will not compile.

After the succesful build of the DLL (window library) you can include the DLL in the VIDE/XIDE project:



And then call the window simply from your Vulcan.NET/X# code:

```
protected virtual method OnStartup( e as StartupEventArgs ) as void
//  local oWindow           as TestFormInherited
    local oWindow           as TestForm

    super:OnStartup(e)
    AppDomain.CurrentDomain:UnhandledException += AppDomainUnhandledException

//  oWindow := TestFormInherited{}
    oWindow := TestForm{}
    oWindow:DataContext := TestFormInheritedViewModel{}
    oWindow:Show( )

    return
```

But beware: **you cannot inherit from a XAML window**! This simply not works as the framework tries to find a XAML for your inherited class.

Written march, 06 2016 by Wolfgang Riedmann wolfgang@riedmann.it