What's New in VFPX 2022

Doug Hennig
Stonefield Software Inc.
Email: doug@doughennig.com
Corporate Web sites: stonefieldquery.com
stonefieldsoftware.com
Personal Web site : DougHennig.com
Blog: DougHennig.BlogSpot.com
Twitter: DougHennig

The last session looking at new VFPX projects was at Southwest Fox in 2018, so it's time for an update. There are at least 20 new projects since then, so this session will cover as many of them as time permits.

Introduction

Since its inception, VFPX has been an incredible resource for VFP developers. It provides free, production-quality code from some of the best VFP developers on the planet over its 135 separate projects (as of July 2022). I personally use 30 of these projects in my development environment and applications and am looking forward to using some of the new projects added in the last couple of years as the need arises.

Introduction to VFPX

If you're new to VFPX, this section is for you.

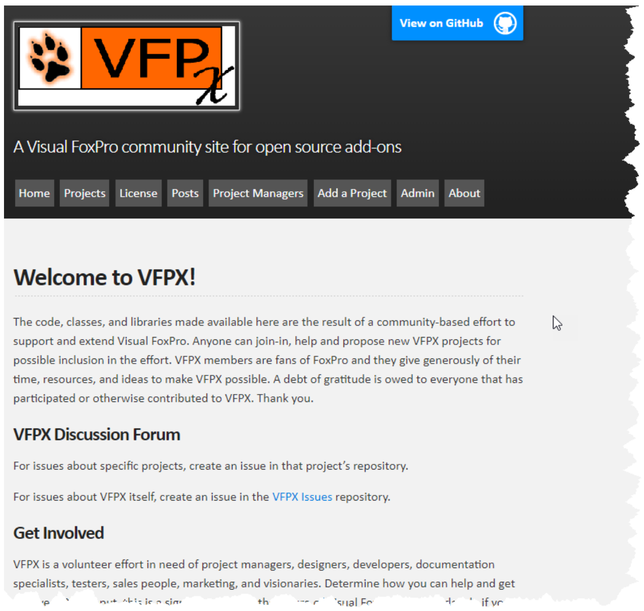
VFPX is located on GitHub, a Microsoft-owned site for Git repositories. Git is a distributed version control system, or DVCS, that's extremely popular with developers world-wide. This document isn't intended to be exhaustive documentation for Git or GitHub; there are many sources of information on both, including https://docs.github.com. Fernando Bozzo, who has contributed numerous VFPX projects, has some excellent Git information at https://github.com/fdbozzo/git_training. Scott Hanselman's blog post "The Squishy Side of Open Source" (https://www.hanselman.com/blog/the-squishy-side-of-open-source) has some good ideas for those new to open source.

Let's start with GitHub, since you don't actually need Git to work with any VFPX project.

GitHub

VFPX is located at https://github.com/vfpx. However, there isn't much to see there: just a list of repositories in reverse chronological order by last update. A better URL is https://vfpx.github.io, the GitHub Pages site for VFPX (GitHub Pages provide a web site for a project). An even better URL is http://vfpx.org, which always redirects to the current location of VFPX.

Whichever URL you use, the VFPX home page (**Figure 1**) is the first thing you see.



Welcome to VFPX!

The code, classes, and libraries made available here are the result of a community-based effort to support and extend Visual FoxPro. Anyone can join-in, help and propose new VFPX projects for possible inclusion in the effort. VFPX members are fans of FoxPro and they give generously of their time, resources, and ideas to make VFPX possible. A debt of gratitude is owed to everyone that has participated or otherwise contributed to VFPX. Thank you.

VFPX Discussion Forum

For issues about specific projects, create an issue in that project's repository.

For issues about VFPX itself, create an issue in the VFPX Issues repository.

Get Involved

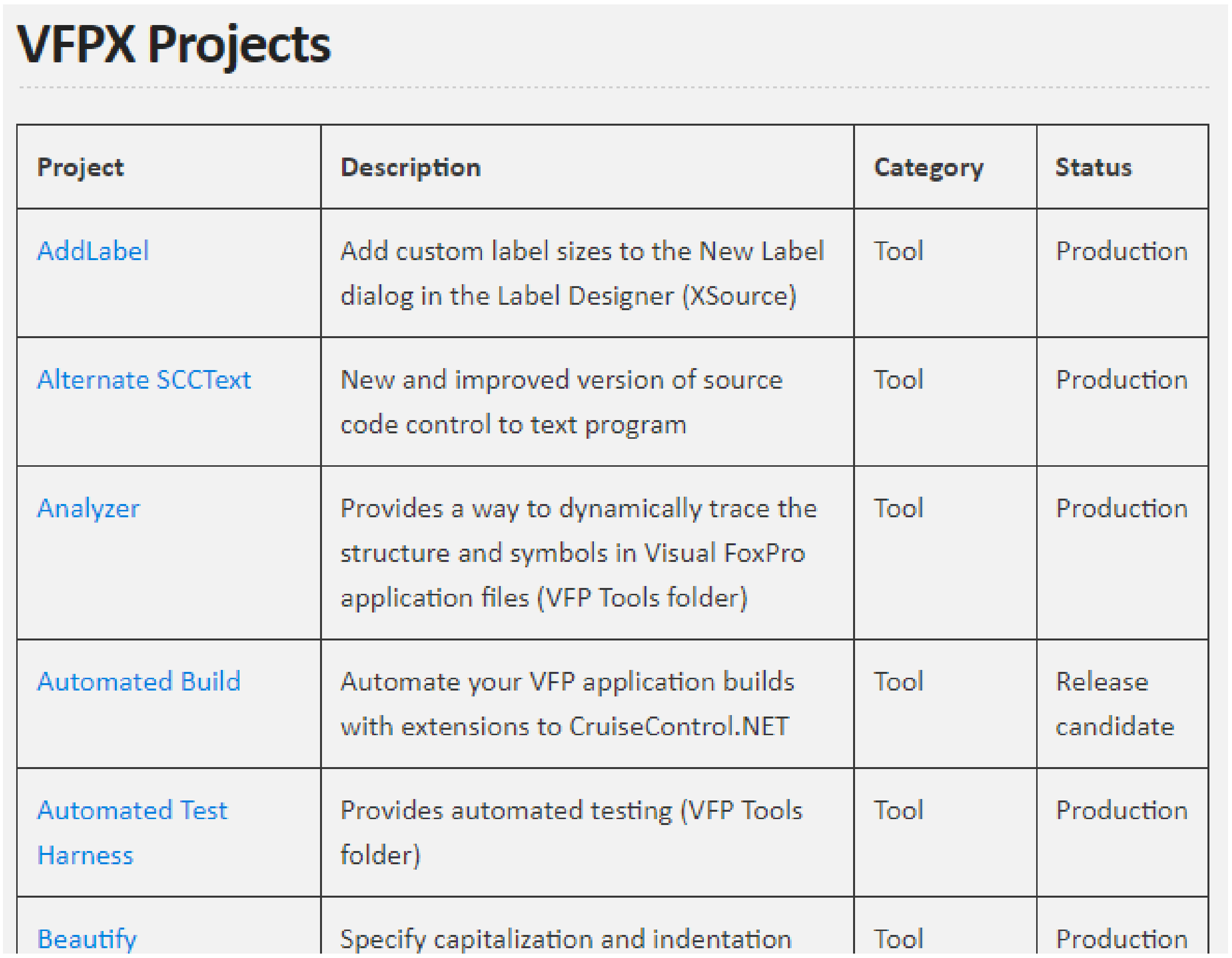
VFPX is a volunteer effort in need of project managers, designers, developers, documentation specialists, testers, sales people, marketing, and visionaries. Determine how you can help and get

Figure 1. The VFPX home page.

The home page describes what VFPX is, how to get involved, and how to promote it. It also has a menu with the following items:

- Home: a link to the home page.
- Projects: a list of the projects including links to their repositories.
- License: the license all VFPX adhere to unless otherwise stated.
- Posts: news items about VFPX. There's an RSS feed for this page (the "subscribe via RSS" link at the bottom) so you'll be notified when posts are added.
- Project Managers: information for project managers.
- Add a Project: information about how to add a project to VFPX.
- Admin: information for the VFPX administrators (our own documentation).
- About: lists the VFPX administrators.

The Projects page (**Figure 2**) is the most used page: it lists the projects alphabetically and shows a brief description, category ("Tool" means a utility that adds features to the VFP Interactive Development Environment, or IDE, and "Component" means a utility that adds features to your VFP applications), and status (Production, Release Candidate, Beta, Alpha, and Planning). The page also lists some other VFP open-source projects that aren't considered to be part of VFPX. Click the name link to navigate to the repository for the project.



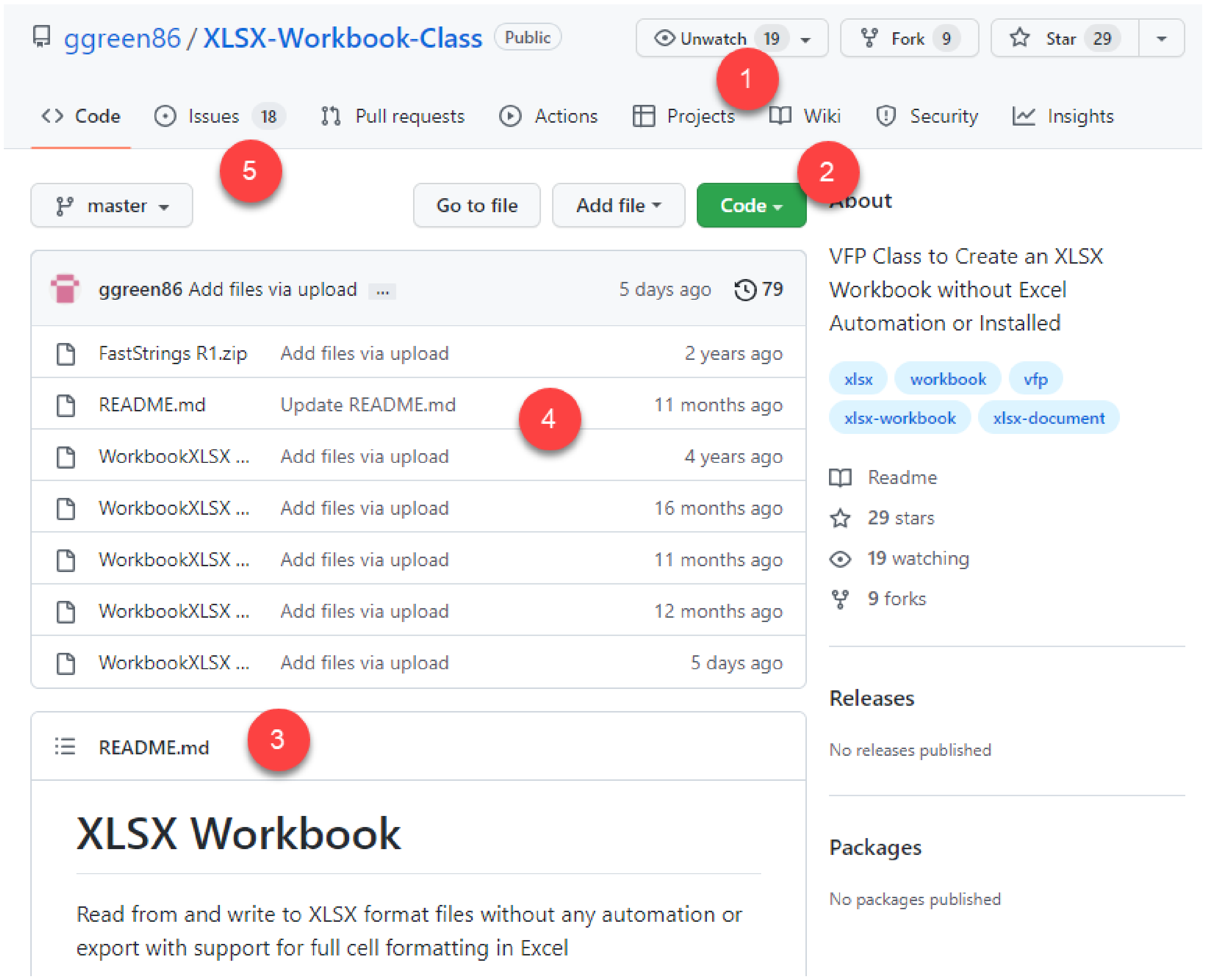
VFPX Projects

Project	Description	Category	Status
AddLabel	Add custom label sizes to the New Label dialog in the Label Designer (XSource)	Tool	Production
Alternate SCCText	New and improved version of source code control to text program	Tool	Production
Analyzer	Provides a way to dynamically trace the structure and symbols in Visual FoxPro application files (VFP Tools folder)	Tool	Production
Automated Build	Automate your VFP application builds with extensions to CruiseControl.NET	Tool	Release candidate
Automated Test Harness	Provides automated testing (VFP Tools folder)	Tool	Production
Beautify	Specify capitalization and indentation	Tool	Production

Figure 2. The VFPX Projects page lists each project by name.

Some projects are hosted on GitHub under the VFPX organization (https://githib.com/vfpx/*ProjectName*) while others are under the author's repository (https://githib.com/*Author*/*ProjectName*).

Figure 3 shows a typical VFPX project repository, in this case the one for the XLXSWorkbook project. As this document is not intended to be complete GitHub documentation, I won't go through everything, just the most common features.



ggreen86 / XLSX-Workbook-Class Public

Unwatch 19 | Fork 9 | Star 29

Code | Issues 18 | Pull requests | Actions | Projects | Wiki | Security | Insights

master | Go to file | Add file | Code

ggreen86 Add files via upload ... 5 days ago 79

FastStrings R1.zip	Add files via upload	2 years ago
README.md	Update README.md	11 months ago
WorkbookXLSX ...	Add files via upload	4 years ago
WorkbookXLSX ...	Add files via upload	16 months ago
WorkbookXLSX ...	Add files via upload	11 months ago
WorkbookXLSX ...	Add files via upload	12 months ago
WorkbookXLSX ...	Add files via upload	5 days ago

README.md

XLSX Workbook

Read from and write to XLSX format files without any automation or export with support for full cell formatting in Excel

About

VFP Class to Create an XLSX Workbook without Excel Automation or Installed

xlsx workbook vfp xlsx-workbook xlsx-document

Readme

29 stars

19 watching

9 forks

Releases

No releases published

Packages

No packages published

Figure 3. A typical VFPX project repository.

- You can get notification via email when the project is updated by clicking the Watch button (#1 in Figure 3) and choosing the desired option. Note that you must be logged in with a GitHub account to turn on watching.
- To install the project on your system, you can either download the project as a zip file or clone the repository, which requires that Git be installed on your machine (#2 in Figure 3).
- The project's README.md file is a Markdown file (Markdown is similar to HTML but with simplified syntax) that GitHub automatically renders on the page (#3 in Figure 3) so it acts as the welcome page and in many case, provides the documentation for the project as well.
- The source list (#4 in Figure 3) shows the folders and files that make up the project. Some projects, such as this one, are just a zip file of source. Most, however, are individual files, such as PRGs, SCXs, and so on. Click a file to view a page containing its content (most binary files can't be viewed, but some, such as PDFs can) and its version history.

- To report a bug, ask a question, or make a feature request, click the Issues tab (#5 in Figure 3) and create an issue. This is much preferred over sending an email to the project manager or posting something on a forum because it allows other users to see and comment on the issue as well.

Git

If you want to clone a project rather than just downloading it, you need to install Git. You can get Git from a variety of sources, including:

- The main source for Git is at https://git-scm.com/downloads. Choose the Windows version for VFPX projects.
- Git is basically a command-line utility, so there are numerous visual interfaces available, including TortoiseGit (https://tortoisegit.org) and SourceTree (https://www.sourcetreeapp.com). Some of these automatically install Git if it isn't already installed, others require you install Git yourself.
- Some people like to use GitHub Desktop (https://desktop.github.com), which is a visual interface for GitHub.

To clone a repository, open a command window somewhere and type:

```
git clone RepositoryURL folder
```

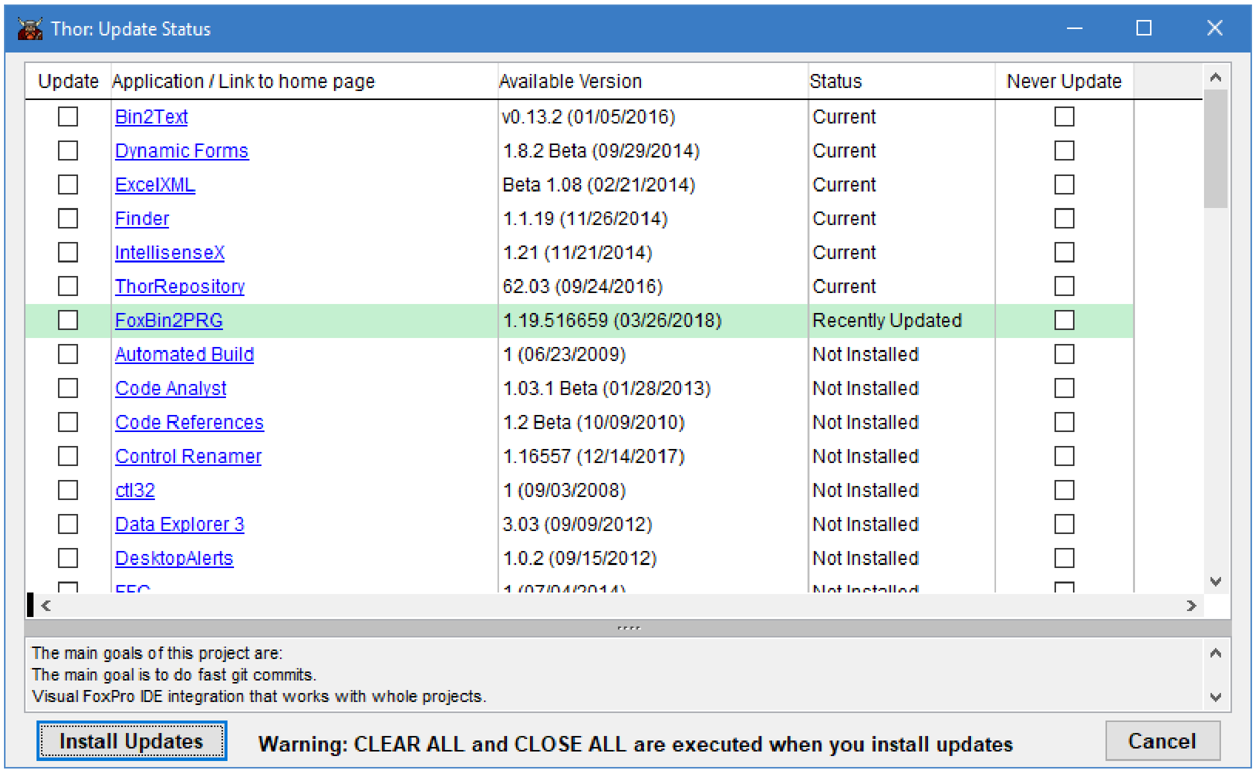
where *RepositoryURL* is the URL displayed when you click the Code button in the repository and *folder* is the folder in which to clone the repository (the folder doesn't have to exist). Alternatively, you can use a visual tool to clone the repository. For example, if you use TortoiseGit, you can simply right-click some folder in File Explorer, choose Git Clone, and enter the URL and folder in the dialog that appears.

As I mentioned earlier, I'm not going to go into detail on how to use Git, as there are a lot of resources available to learn Git.

Thor Check for Updates

Another way to install a project is by installing Thor and then using its Check for Updates (CFU) function (**Figure 4**). Simply put a checkmark in the Update column for those projects you want to install or update and click Install Updates.

Thor CFU downloads files from a folder on a web site unrelated to GitHub. Since project managers don't have write access to that site, they have to submit update files to the VFPX administrators to put them in the proper folder. This mechanism is kind of clunky so it's been improved by adding support for updates to be downloaded directly from a project's GitHub repository. However, not many projects have been updated to do that yet, so the version number you see in the Thor CFU dialog doesn't necessarily match the one in the project's repository; the GitHub files could be newer than the ones in the VFPX site.



Thor: Update Status

Update	Application / Link to home page	Available Version	Status	Never Update
☐	Bin2Text	v0.13.2 (01/05/2016)	Current	☐
☐	Dynamic Forms	1.8.2 Beta (09/29/2014)	Current	☐
☐	ExcelXML	Beta 1.08 (02/21/2014)	Current	☐
☐	Finder	1.1.19 (11/26/2014)	Current	☐
☐	IntellisenseX	1.21 (11/21/2014)	Current	☐
☐	ThorRepository	62.03 (09/24/2016)	Current	☐
☐	FoxBin2PRG	1.19.516659 (03/26/2018)	Recently Updated	☐
☐	Automated Build	1 (06/23/2009)	Not Installed	☐
☐	Code Analyst	1.03.1 Beta (01/28/2013)	Not Installed	☐
☐	Code References	1.2 Beta (10/09/2010)	Not Installed	☐
☐	Control Renamer	1.16557 (12/14/2017)	Not Installed	☐
☐	ctl32	1 (09/03/2008)	Not Installed	☐
☐	Data Explorer 3	3.03 (09/09/2012)	Not Installed	☐
☐	DesktopAlerts	1.0.2 (09/15/2012)	Not Installed	☐

The main goals of this project are:
The main goal is to do fast git commits.
Visual FoxPro IDE integration that works with whole projects.

Install Updates
Warning: CLEAR ALL and CLOSE ALL are executed when you install updates
Cancel

Figure 4. Thor's Check for Updates function can install and update many VFPX projects.

Now let's dig into what's new in VFPX.

What's new in VFPX

Here are the projects added to VFPX as of this writing since my "VFPX 2018 Edition" presentation at Southwest Fox 2018, in order of date added:

- FoxMock: a mocking framework for testing VFP applications.
- UTCDatetime: a class to handle UTC dates and times.
- FoxFaker: a library that generates fake data for testing purposes.
- fxReports: a framework for sharing custom report features. We won't look at this project because it's extensive documented and is a little too complex for this discussion.
- overHere: a wrapper for Here.com to easily integrate mapping and geolocation services and data into VFP applications. We won't look at this project because I don't have a Here.com account.
- OOP Reports: an object-oriented wrapper for VFP reports.
- Multi-Select Combobox: a combobox-style control for selecting multiple values from a list and storing as a single, character-separated string.

- Tooltip: a control providing customized tooltips, supporting multiple lines of text and balloon-style display, both automatically and on demand.
- FoxRegEx: a VFP wrapper for the VBScript RegEx object, which provides regular expression processing.
- FoxStack: a class which implements a stack data structure in VFP.
- FoxStringBuilder: a VFP implementation of the .NET StringBuilder class.
- FoxQueue: a class which implements a queue data structure in VFP.
- BlackFox: a class that abstracts other language constructs such as JSON notation and HTTP protocol.
- ZintBarcode: a wrapper for the Zint Barcode Generator library.
- Ribbon: a Microsoft Office 365-like ribbon control for VFP forms.
- SFMail: a library to send emails from VFP applications, including support for Modern Authentication.
- VFP Editors: replacements for the VFP Form, Class, Menu, and Table Designers plus the program editor and Project Manager. We won't look at this project because I try to avoid installing ActiveX controls on my system.
- Error Handler: a highly configurable and customizable error handler for any VFP application.
- FoxEnv: a library for storing secret data, such as database credentials and API keys.
- Object Explorer: an explorer form to view members of VFP objects, even at runtime.

In addition, two projects were "moved." Matt Slay passed away in 2021. He was a great contributor to the Fox community and VFPX, and a great guy, and is sadly missed. His two projects, GoFish and Dynamic Forms, were under his own GitHub account, which means no one can make changes. So, we forked those repositories and changed the VFPX project list to point to the forked ones rather than Matt's. This allows his legacy to continue by permitting other people to contribute to these projects.

In addition to new projects, lots of existing projects have been enhanced since 2018. For example, Greg Green has been doing continuous improvements to XLSXWorkbook, a project that provides a class to read from and write to Microsoft Excel documents. Lutz Scheffler has fixed numerous bugs and made improvements in several projects, including PEM Editor, Thor Repository, and FoxBin2PRG. The Thor Check for Updates process has been tweaked to be more reliable.

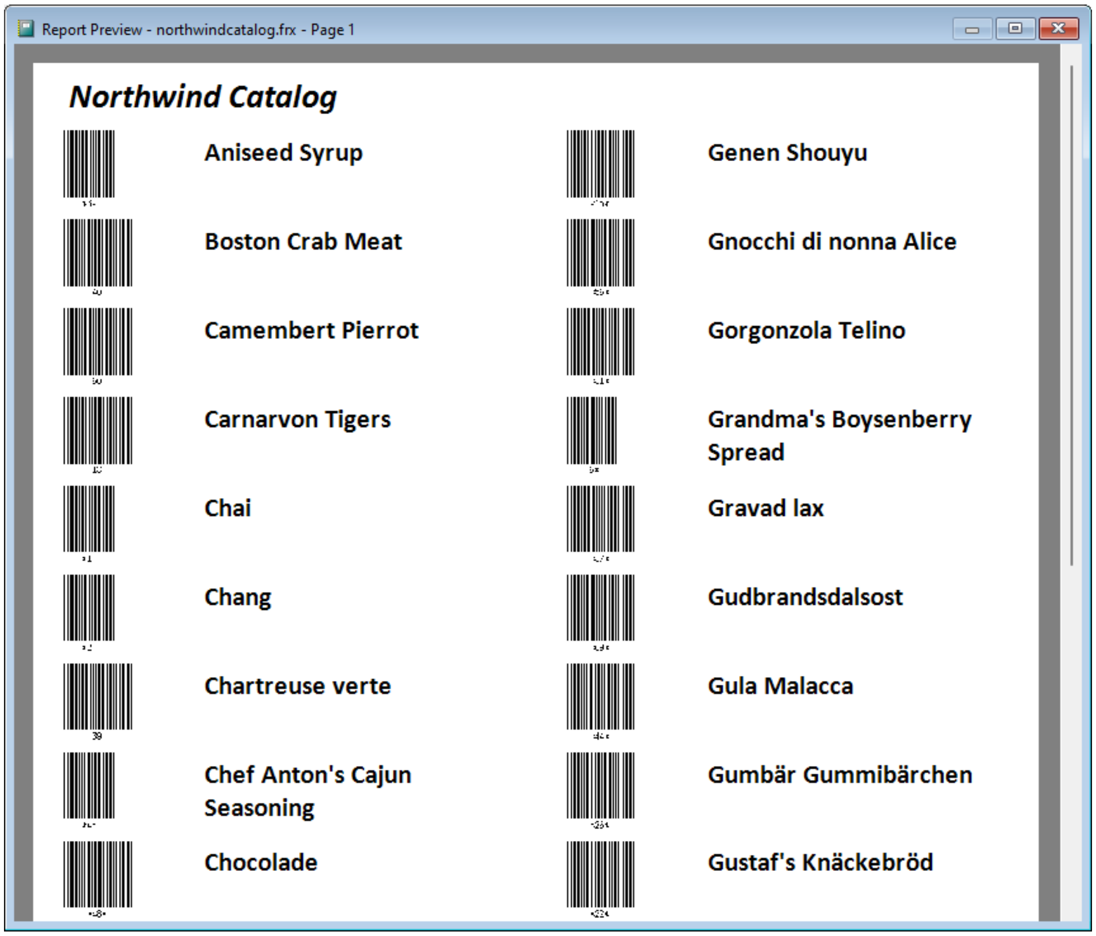
We'll start by looking at some of the smaller, simpler projects and work our way up to the larger, more complex ones. I won't reproduce any of the project documentation but instead give my take on each project.

ZintBarcode

This project by Antonio Lopes is a VFP wrapper for Zint, an open-source barcode generator library. It can create both 1-D and 2-D barcodes, including QR codes, and supports overlaying the barcodes with other images. The project has detailed documentation and several examples. **Listing 1** and **Figure 5** show the code for and the resulting output of a simple example that produces barcodes for products in the Northwind sample database.

Listing 1. ZintBarcode can create barcodes with just a few lines of code.

```
DO (JUSTPATH(SYS(16)) + "\..\src\zintbarcode.prg")

LOCAL ZB AS ZintBarcode
LOCAL ZE AS ZintEnumerations

m.ZB = CREATEOBJECT("ZintBarcode")
m.ZE = CREATEOBJECT("ZintEnumerations")

m.ZB.SetSymbology(m.ZE.Barcode_code39)

IF !DBUSED("Northwind")
   OPEN DATABASE (HOME(0) + "\Samples\Northwind\Northwind") NOUPDATE SHARED
ENDIF

SELECT cpl.ProductId, ;
       cpl.ProductName, ;
       m.ZB.Imagefile(TRANSFORM(cpl.ProductId)) AS barcode ;
   FROM Northwind!Current_Product_List cpl ;
   INTO CURSOR curReport

REPORT FORM (JUSTPATH(SYS(16)) + "\northwindCatalog") PREVIEW
```



Northwind Catalog

Aniseed Syrup	Genen Shouyu
Boston Crab Meat	Gnocchi di nonna Alice
Camembert Pierrot	Gorgonzola Telino
Carnarvon Tigers	Grandma's Boysenberry Spread
Chai	Gravad lax
Chang	Gudbrandsdalsost
Chartreuse verte	Gula Malacca
Chef Anton's Cajun Seasoning	Gumbär Gummibärchen
Chocolade	Gustaf's Knäckebröd

Figure 5. Barcodes generated by ZintBarcode.

For more complex barcodes, there are many properties you can set, such as Symbology, WhitespaceHeight, Scale, and DotSize.

There are a couple of other barcode generating VFPX projects: FoxBarCode and FoxBarCodeQR. One difference between them and ZintBarcode is that the latter requires a couple of supporting files, vfp2c32.fll and zint.dll, that must be distributed with your application. It also requires System.app, the main file from the GDIPlusX VFP project, if you create overlays for barcodes.

I use FoxBarCode and FoxBarCodeQR in Stonefield Query, my company's main product, so I haven't had a need to use ZintBarcode. However, if you need barcodes, check them both out so you can decide which works better for you.

FoxQueue and FoxStack

These two projects, both by Irwin Rodríguez, provides queue and stack data structures as an alternative to using arrays or collections. Both are standalone one-PRG classes.

FoxQueue is a wrapper for an array with Enqueue (add to the queue), Dequeue (remove from the queue in first-to-last order), Extract (remove from the queue in last-to-first order), Peek (retrieve the item at the top of the queue without removing it), Clear (remove all items), and Count (return the number of items) methods. **Listing 2** shows an example of these methods. Note that queue elements can be different data types if necessary.

Listing 2. FoxQueue provides a queue-like wrapper for an array.

```
set procedure to FoxQueue

loQueue = createobject('FoxQueue')

* Add items to the queue.

loQueue.Enqueue('Tamar')
loQueue.Enqueue('Rick')
loQueue.Enqueue('Doug')
loQueue.Enqueue(2022)
loQueue.Enqueue(.T.)

* Dequeuing removes them in first to last order.

clear
lnCount = loQueue.Count()
? 'Dequeuing:'
? transform(lnCount) + ' items'
for lnI = 1 to lnCount
   ? loQueue.Dequeue()
next lnI
lnCount = loQueue.Count()
? transform(lnCount) + ' items'

* Extracting removes them in last to first order.

loQueue.Enqueue('Tamar')
loQueue.Enqueue('Rick')
loQueue.Enqueue('Doug')
loQueue.Enqueue(2022)
loQueue.Enqueue(.T.)

lnCount = loQueue.Count()
?
? 'Extracting:'
? transform(lnCount) + ' items'
for lnI = 1 to lnCount
   ? loQueue.Extract()
next lnI
lnCount = loQueue.Count()
? transform(lnCount) + ' items'

* Peek retrieves the top item without removing it.

loQueue.Enqueue('Tamar')
loQueue.Enqueue('Rick')
loQueue.Enqueue('Doug')
loQueue.Enqueue(2022)
loQueue.Enqueue(.T.)
lnCount = loQueue.Count()
?
? 'Peeking:'
? transform(lnCount) + ' items'
? loQueue.Peek()
lnCount = loQueue.Count()
? transform(lnCount) + ' items'

* Clear removes all items.

?
? 'Clearing:'
? transform(lnCount) + ' items'
? loQueue.Clear()
lnCount = loQueue.Count()
? transform(lnCount) + ' items'
```

FoxStack is also a wrapper for an array, but processes elements in last-in, first-out (LIFO) order. It has Push (add to the stack), Pop (remove from the stack in LIFO order), Peek (retrieve the item at the top of the stack without removing it), Empty (return .T. if there are no items on the stack), and Size (return the number of items) methods. **Listing 3** shows an example of these methods. Note that stack elements can be different data types if necessary.

Listing 3. FoxStack also provides a wrapper for an array but using stack structure.

```
set procedure to FoxQueue&FoxStack\FoxStack

loStack = createobject('FoxStack')

* Add items to the stack.

loStack.Push('Tamar')
loStack.Push('Rick')
loStack.Push('Doug')
loStack.Push(2022)
loStack.Push(.T.)

* Popping removes them in last-in, first-out (LIFO) order.

clear
lnCount = loStack.Size()
? 'Popping:'
? transform(lnCount) + ' items'
? 'The stack is ' + iif(loStack.Empty(), 'empty', 'not empty')
for lnI = 1 to lnCount
   ? loStack.Pop()
next lnI
lnCount = loStack.Size()
? transform(lnCount) + ' items'
? 'The stack is ' + iif(loStack.Empty(), 'empty', 'not empty')

* Peek retrieves the top item without removing it.
```

```
loStack.Push('Tamar')
loStack.Push('Rick')
loStack.Push('Doug')
lnCount = loStack.Size()
?
? 'Peeking:'
? transform(lnCount) + ' items'
? loStack.Peek()
lnCount = loStack.Size()
? transform(lnCount) + ' items'

* Search finds an item in the stack.

?
? 'Searching:'
? loStack.Search('Rick')
```

I use my own stack class for “go back” functionality in Explorer-like forms: as the user navigates through the TreeView, the nodes are pushed onto the stack, and when they click Back button, the nodes are popped off the stack and reselected. So, I haven’t needed to use either of these projects, but would if I needed this functionality.

FoxEnv

This project, also by Irwin Rodríguez, a library for reading name/value pairs from a .env file. It can be used for storing configuration settings or secret data, such as database credentials and API keys. It’s a standalone one-PRG class.

To load the values, simply DO FoxEnv with the path to the .env file. The values are loaded as properties of the Env member of _SCREEN, so the names must be valid property names. FoxEnv doesn’t have a function to save values, so you must write out to the .env file yourself. It also doesn’t support encryption, so you must encrypt and decrypt values yourself.

Some things FoxEnv does support are:

- Multi-line values: they must start and end with triple quotes.
- Interpolation: substitutes one value for another. Surround the text to be substituted with “${“ and “}.”
- Comments: use text like “Name = Value # Comment.”
- Automatic data type conversion.

All of these are shown in **Listing 4**.

Listing 4. FoxEnv gets settings, such as connection strings or credentials, from a .env file.

```
* Create an env file.
```

```
text to lcValues noshow
USERNAME = dhennig
PASSWORD = mypassword
EMAIL = ${USERNAME}@stonefieldquery.com # This shows interpolation
USESSL = "true"
PORT = "587"
PRIVATEKEY = """
---- BEGIN SSH2 PUBLIC KEY ----
AAAAB3NzaC1yc2EAAAABJQAAAQB/nAmOjTmezNUDKYvEeIRf2YnwM9/uUG1d0BYs
c8/tRtx+RGi7N2lUbp728MXGwdnL9od4cItzky/zVdLZE2cycOa18xBK9cOWmcKS
0A8FYBxEQWJ/q9YVUgZbFKfYGaGQxsER+A0w/fX8ALuk78ktP31K69LcQgxIsl7r
NzxsoOQKJ/CIxOGMMxczYTiEoLvQhapFQMs3FL96didKr/QbrfB1WT6s3838SEaX
fgZvLef1YB2xmfhbT9OXFE3FXvh2UPBfN+ffE7iiayQf/2XR+8j4N4bW30DiPtOQ
LGUrH1y5X/rpNZNlWW2+jGIxqZtgWg7lTy3mXy5x836Sj/6L
---- END SSH2 PUBLIC KEY ----
"""
endtext
strtofile(lcValues, 'FoxEnv\settings.env')

* Read and display the settings.

do FoxEnv\FoxEnv with 'FoxEnv\settings.env'
clear
? _screen.Env.UserName
? _screen.Env.Password
? _screen.Env.Email
? _screen.Env.UseSSL
? _screen.Env.Port
? _screen.Env.PrivateKey
```

FoxEnv is a lighter weight alternative to other similar classes, such as the Settings class in the My VFPX project. A nice addition to FoxEnv would be adding built-in encryption.

FoxStringBuilder

This is yet another project from Irwin Rodríguez. It's a VFP implementation of the .NET StringBuilder class. Because concatenating large strings using "+" can cause a lot of memory usage and the resulting need for garbage collection, C# developers use the Append method of StringBuilder instead. FoxStringBuilder does something similar in VFP. **Listing 5** shows an example.

Listing 5. FoxStringBuilder is a VFP implementation of the .NET StringBuilder class.

```
set procedure to FoxStringBuilder\FoxStringBuilder

* Append.

clear
loBuilder = createobject('FoxStringBuilder')
loBuilder.Append('Hello')
loBuilder.Append(' World!')
? loBuilder.ToString()
```

```
* FoxStringBuilder supports appending in Init.

loBuilder = createobject('FoxStringBuilder', 'Hello World!')

* It supports data type conversion.

loBuilder.Append('Vfp Rules!')
loBuilder.Append(.T.)
loBuilder.Append(1985)
? loBuilder.ToString()

* You can append another FoxStringBuilder object.

loBuilderObjA = createobject('FoxStringBuilder')
loBuilderObjB = createobject('FoxStringBuilder')
loBuilderObjA.Append('Hello')
loBuilderObjB.Append(loBuilderObjA)
loBuilderObjB.Append(' World!')
? loBuilderObjB.ToString()

* Delete and DeleteCharAt.

loBuilder.Delete(6, 12)
? loBuilder.ToString()
loBuilder.DeleteCharAt(6)
? loBuilder.ToString()

* IndexOf and LastIndexOf.

loBuilder = createobject('FoxStringBuilder')
loBuilder.Append('Hello World!')
? loBuilder.IndexOf('o')
? loBuilder.LastIndexOf('o')

* Insert.

loBuilder.Insert(5, ' Dear')

* Replace.
loBuilder = createobject('FoxStringBuilder')
loBuilder.Append('Hello World!')
? loBuilder.Replace(5, 6, ' My Dear ')

* Reverse.

? loBuilder.Reverse()
```

Although FoxStringBuilder supports automatic data type conversion so you don't have to use TRANSFORM(), one feature of the .NET version it doesn't support is format strings. That allows you to use something like this:

```
loBuilder.AppendFormat("There are {0} records in {1}", reccount(), alias())
```

rather than the more awkward:

```
loBuilder.Append("There are ")
loBuilder.Append(reccount())
loBuilder.Append(" records in ")
loBuilder.Append(alias())
```

or using TRANSFORM() on RECCOUNT().

FoxStringBuilder has some abilities beyond simply concatenating strings so you might find it useful for some tasks. Having an AppendFormat method, though, would make it much more useful.

UTCDatetime

This is another project from Antonio Lopes that handles UTC times. It comes as both an APP file and the source code for the APP. You can just deploy UTC.app with your application. If you want to use the source instead, you'll also need to install iCal4VFP, another VFPX project from Antonio, since UTCDatetime requires several of the PRGs in that project.

UTCDatetime uses IANA time zone codes, which you can download from https://www.iana.org/time-zones if you want a list of time zones around the world (such as to display in a user interface).

To use UTCDatetime, DO UTC.app to add a UTCDatetime object named UTC to _SCREEN. This object has several methods for handling UTC date and time values, including:

- SetTimezone(Timezone): sets the time zone to the specified IANA code, such as "America/Winnipeg."
- Now: returns the current UTC time.
- UTCTime([DateTime] [, Timezone]): returns the UTC time for the specified local value. The default for DateTime is the current time and the default for Timezone is the current time zone.
- LocalTime([DateTime] [, Timezone]): returns the local time for the specified UTC value. The default for DateTime is the current time and the default for Timezone is the current time zone.
- GetTimeDifference(DateTime1, Timezone1, DateTime2, Timezone2): returns the difference in seconds between two times.
- TTOC(LocalTime [, Timezone] [, Options]): returns the specified local time for the specified time zone, with options including format to use (ISO8601, YYYY-MM-DDTHH:MM:SS±HH:MM, or RFC2822, Wkd, DD Mon YYYY HH:MM:SS±HHMM) and whether or not to include the time zone name.
- CTOT(UTCTimeString): converts a time string in ISO8601 or RFC2822 format to a UTC time.

Listing 6 shows examples using these methods.

Listing 6. UTCDatetime provides methods to handle UTC times.

```
do UTCDatetime\UTC.app
clear

_screen.UTC.SetTimezone('America/Winnipeg')

ltUTCTime = _screen.UTC.Now()

? 'UTC:', ltUTCTime
? 'Local:', _screen.UTC.LocalTime(ltUTCTime)
? 'In Vancouver:', _screen.UTC.LocalTime(ltUTCTime, 'America/Vancouver')
? 'In Toronto:', _screen.UTC.LocalTime(ltUTCTime, 'America/Toronto')
? 'UTC for 10/01/2022 10:05:25 local:', ;
   _screen.UTC.UTCTime(datetime(2022, 10, 1, 10, 5, 25))

?
? 'TTOC: ISO8601:', _screen.UTC.TTOC(ltUTCTime)
? 'TTOC, display time zone name:', _screen.UTC.TTOC(ltUTCTime, , 1)
? 'TTOC, RFC2822:', _screen.UTC.TTOC(ltUTCTime, , 0x4000)

?
? 'CTOT, RFC2822:', _screen.UTC.CTOT('Mon, 27 Jun 2022 10:15:00 -0600')
? 'CTOT, ISO8601:', _screen.UTC.CTOT('2022-06-27T10:15:00-06:00')

lcAirline     = 'Air France'
lcFlight      = '22'
ltDeparture   = {^2022-03-16 08:36:00}
lcDepAirport  = 'Paris - De Gaulle'
lcDepTimezone = 'Europe/Paris'
ltArrival     = {^2022-03-16 11:45:00}
lcArrAirport  = 'New York - JFK'
lcArrTimezone = 'America/New_York'
lnDuration    = _screen.UTC.GetTimeDifference(ltDeparture, lcDepTimezone, ;
   ltArrival, lcArrTimezone)
?
? lcAirLine + ' ' + lcFlight
? '    departing ' + lcDepAirport + ' at ' + transform(ltDeparture)
? '    arriving '  + lcArrAirport + ' at ' + transform(ltArrival)
? '    duration: ' + transform(lnDuration/3600, '99.99')
```

This is definitely a useful component if you need to work with UTC times in your application.

BlackFox

Yet another project by Irwin Rodríguez, this one is quite interesting, if not from a practical viewpoint, at least from a theoretical one. The project description states that “BlackFox is a Command Based Interpreter embedded in Visual FoxPro.” After you DO BlackFox.app, simply storing a command to the BlackFox property of _SCREEN parses and executes it. In a way, it provides the ability to extend the VFP language, albeit a little clumsily. Here’s a made-up example:

```
_screen.BlackFox = 'DOWNLOAD http://mysite/somefile TO C:\Folder\SomeFile'
```

and that file now exists. Or:

```
_screen.BlackFox = 'SEND TO dhennig@stonefield.com SUBJECT [Test Message] ' + ;
   'BODY [Here is a message.] ATTACHMENT SomeFile.txt'
```

and an email is sent. These examples don't work right now because BlackFox currently only supports JSON and HTTP commands, but the design of BlackFox is open to adding whatever parsing you wish. **Listing 7** shows an example of parsing and executing commands.

Listing 7. BlackFox provides a command interpreter for VFP.

```
set path to BlackFox
do BlackFox.app
clear

text to _vfp.BlackFox noshow textmerge pretext 7
  # SEND HTTP USING GET METHOD
  HTTP GET oHTTP
    URL 'https://jsonplaceholder.typicode.com/todos/1'
    ADD HEADER 'Content-Type' with 'application/json'
endtext
? 'HTTP GET example:', oHTTP.responseText

text to lcBody noshow
  {
    "Id": 78912,
    "Customer": "Jason Sweet",
    "Quantity": 1,
    "Price": 18.00
  }
endtext
text to _vfp.BlackFox noshow textmerge pretext 7
  # SEND HTTP USING POST METHOD
  HTTP POST oHTTP
    URL 'https://reqbin.com/echo/post/json'
   BODY '<<lcBody>>'
   ADD HEADER 'Content-Type' with 'application/json'
endtext
?
? 'HTTP POST example:', oHTTP.responseText

text to _vfp.BlackFox noshow textmerge pretext 7
  CREATE JSON OBJECT oJSON FROM lcBody
endtext
?
? 'JSON parsing:', oJSON.Customer
```

As I mentioned, this may not have much practical use yet (there are other HTTP and JSON libraries for VFP), but keep an eye on this project, or even better, if you like the idea of extending the VFP language, contribute parsing code for other syntax.

FoxRegEx

This project, yet another one by Irwin Rodríguez, is a VFP wrapper for the VBScript RegEx object, which provides regular expression processing. It comes as a single PRG, FoxRegEx.prg, you add to your application's project. Instantiate the FoxRegEx class in that PRG, set properties, and call the Execute method to perform regular expression parsing. It supports returning an object or creating a cursor of matches. Call Test to return .T. if a pattern matches a string or Replace to do a substring replacement. FoxRegEx also includes some built-in validators so you don't have to figure out a pattern string for common needs, such a valid email address format. **Listing 8** shows examples using FoxRegEx.

Listing 8. FoxRegEx provides a wrapper for regular expression processing.

```
loRegEx = newobject('FoxRegEx', 'FoxRegEx\FoxRegEx.prg')
clear

* Test does pattern validation.

loRegEx.Pattern = 'vfp'
? 'Test:', loRegEx.Test('vfp Rocks!')

* Create a cursor of matches.

loRegEx.Global     = .T.
   && match all occurrences
loRegEx.IgnoreCase = .T.
loRegEx.Pattern    = '\b\w+\b'
loRegEx.UseCursor  = .T.
loRegEx.CursorName = 'Matches'
loRegEx.Session    = 1
   && default datasession
lnCount = loRegEx.Execute('the mouse and the cat')
if lnCount > 0
   select Matches
   browse
endif lnCount > 0

* Create a collection of matches.

loRegEx.UseCursor = .F.
loMatches = loRegEx.Execute('the mouse and the cat')
?
? 'Matches:'
for each loItem in loMatches
   ? loItem.Value
next loItem

* Replacement.

loRegEx.Pattern = 'cat'
lcText = loRegEx.Replace('the mouse and the cat', 'cheese')
?
? 'Replacing "cat" with "cheese":', lcText
```

```
* Built-in validators.

?
? 'IsURL:',      loRegEx.isURL('https://github.com/Irwin1985/FoxRegEx')
? 'IPv4:',       loRegEx.isIPv4('192.168.0.1')
? 'IPv6:',       loRegEx.isIPv6('2001:db8:0:1:1:1:1:1')
? 'Email:',      loRegEx.isEmail('doug@doughennig.com')
? 'Date:',       loRegEx.isDate('1985-11-15', 'YYYY-mm-dd')
? 'Visa:',       loRegEx.isCreditCard('4872272392854644', 'Visa')
? 'MasterCard:', loRegEx.isCreditCard('5247233849623284', 'MasterCard')
```

I use the VBScript RegEx class in some of my applications so I haven't needed this library, but the built-in validators seem like they'd be very useful if you need them.

FoxMock

This project from Christof Wollenhaupt allows you to create mock objects for testing purposes. Why use mock objects? Suppose you're testing a class and that class collaborates with another class that reads from or writes to a database. Doing so goes beyond the purpose of testing the class, so using a mock object that doesn't really access the database but only acts like it does is the preferred approach. There are numerous mock frameworks in the .NET and other development worlds; this is the only one I'm aware of for VFP.

Although it doesn't require it, FoxMock works best with FoxUnit, another VFPX project that provides a testing framework for VFP. In the Setup procedure for tests, create a FoxMock object:

```
public mock
mock = newobject('foxMock', 'foxMock.prg')
```

In the TearDown procedure, release all FoxMock object references:

```
mock.CleanUp()
release mock
```

In a test, create a new mock using code like this, which defines a property named lDebugMode and sets its value to .T.:

```
loObj = mock.New.Property("lDebugMode").Is('.T.')
```

This defines a method named IsAdmin that returns .F. when called:

```
loObj = mock.New.CallTo('IsAdmin').Return('.F.')
```

Notice the chained, or "fluent," method calls. This is an interesting technique I've seen lots of times in C# but never in VFP. Here's an example creating a mock that returns different values each time it's called:

```
loRef = mock.new.Method('Test').Return('1').Then.Return('2').Then.Return('3')
```

Although there isn’t currently any documentation for FoxMock, the FoxUnit tests included in the project show the various ways mocks can be set up.

FoxFaker

This project, yet another from Irwin Rodríguez, generates fake data. FoxFaker was intended to be used with FoxUnit and FoxMock for testing but can be used, for example, to fill tables with fake data. This can be useful for training new users or demos; rather than working against a production database with real customer information, the application works against a test database of fake information.

FoxFaker consists of FoxFaker.prg and FactoryBase.txt, which is actually an encoded table that’s written out to a DBF by the FoxFaker class. Instantiate a FoxFaker object:

```
loFaker = newobject('FoxFaker', 'FoxFaker.prg')
```

then call Fake* methods to return fake data (see **Listing 9**).

Listing 9. FoxFaker generates fake data for testing, training, demos, and other purposes.

```
create cursor Contacts (ID I, ;
   FirstName C(20), ;
   LastName C(20), ;
   Title C(40), ;
   Company C(40), ;
   Address C(40), ;
   City C(20), ;
   Region C(20), ;
   Country C(30), ;
   PostCode C(10), ;
   Phone C(20), ;
   Email C(40), ;
   Created D, ;
   Comment M)

for lnI = 1 to 20
   append blank
   replace ID with loFaker.FakeNumberBetween(1, 10000), ;
      FirstName with loFaker.FakeFirstName(), ;
      LastName with loFaker.FakeLastName(), ;
      Title with loFaker.FakeJobTitle(), ;
      Company with loFaker.FakeCompany(), ;
      Address with loFaker.FakeStreetAddress(), ;
      City with loFaker.FakeCity(), ;
      Region with loFaker.FakeState(), ;
      Country with loFaker.FakeCountry(), ;
      PostCode with alltrim(loFaker.FakePostCode()), ;
      Phone with loFaker.FakePhoneNumber(), ;
      Email with loFaker.FakeEmail(), ;
      Created with loFaker.FakeDate(), ;
      Comment with loFaker.FakeText(100)
next lnI
```

As you can see, there are a lot of these methods (the README for the project lists all of them). Using this tool is much easier than creating your own fake data for any purpose.

Multi-select combobox

This project by Greg Willcockson provides a combobox-style control for selecting multiple values from a list and storing as a single, character-separated string. It's a single class, cboMultiSel, in a single class library, cboMultiSel.vcx. Simply drop the class on a form or class and set some properties to configure it. **Figure 6** shows the sample form included in the project.

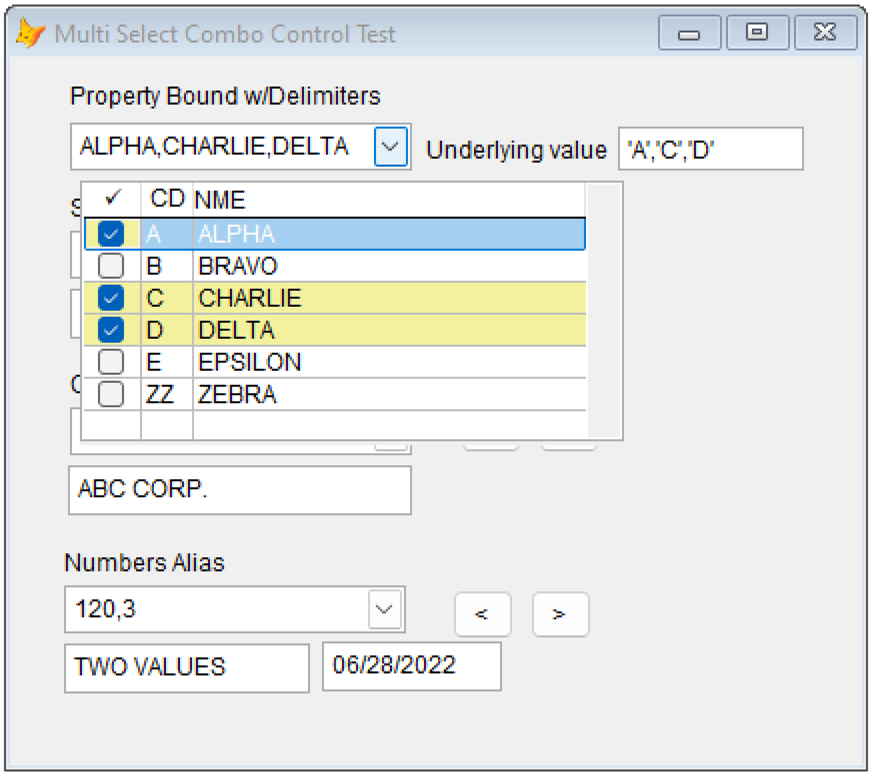


Figure 6. cboMultiSel provides a combobox-like control for selecting multiple values.

The first combobox in the sample form is set up using this code:

```
WITH Thisform.cboMultiSel1
    .RowSourceType = 6  && Fields
    .ControlSource = "Thisform.mySels"
    .ColumnWidths = "25, 200"
    .srcValCol = "CD"
    .srcDispCol = "NME"
    .valDelim = "'"
    * set this property last if setting other properties in code (as above)
    .RowSource = "CSRC.CD, NME"
ENDWITH
```

RowSourceType must be set to 2-Alias, 3-SQL statement, or 6-Fields. ControlSource is set to a property of the form, which the textbox to its right is also bound to. ColumnWidths defines the widths of the columns in the combobox. srcValCol is the expression (usually a column in a cursor) containing the values and srcDispCol is the expression (again, usually a column in a cursor) for the description of those values. valDelim is an optional delimiter to surround selected values with. RowSource is set to the columns in a cursor containing the values and their descriptions. Some other properties you can set are:

- srcColTitls: a comma-delimited list of the column headers to use when the list is dropped down.
- showHdrs: set this to .F. to not display column headers.
- srcColDefs: a two-dimensional array defining the columns. Each row is a column in the list. Column 1 of the array contains the column name in the cursor, column 2 contains the width of the column, column 3 contains the column heading, and column 4 contains the InputMask for the column.

Dropping down the list displays a form with no border containing a grid showing the records in the cursor plus a checkbox column for selecting values. The control supports two-way data binding: selecting values in the grid updates the ControlSource with the values (optionally delimited) and changing the ControlSource updates both the combobox display and the selected items in the grid when the control is refreshed.

I’ve used this component in one application and find it to be very useful for multi-select lists, as Shift- or Ctrl-clicking choices in a long combobox is a pain.

Tooltip

This is another project by Greg Willcockson. It’s a control providing customized tooltips, supporting multiple lines of text and balloon-style display, both automatically and on demand. Like his Multi-Select Combobox, Tooltip is a single class, ctlTip, in a single VCX, ctlTip.vcx.

Drop the class on a form or class (a ctlTip object can provide tooltips for multiple controls, so you likely only need one), set its TipStyle property to 0 (the default) for a standard tooltip window or 1 for a balloon-style display, and call its RegTool method to register any controls you want tooltips displayed for. The parameters for RegTool are:

- A reference to the control to register.
- .T. for a demand style tooltip, .F. for a standard style tooltip. Demand style is more flexible, as you can see from other parameters that only apply to demand style.
- The position of the tooltip: 0 = at the mouse position, 1 = centered in the control, 2 = use the last two parameters for the position (demand style only).
- The tooltip text for this control. You can change the text dynamically by calling SetTxt.
- The title for the tooltip for balloon-style tooltips. You can change the title dynamically by calling SetTitl.

- An icon to display for balloon-style tooltips: 0 = none, 1 = info, 2 = warning, 3 = error. Use 4, 5, or 6, respectively, to show large versions of the icon.
- When to activate the tooltip (demand style only): 0 = mouse enter, 1 = on focus, 2 = both, 3 = none (you must call ShowTip() to display it).
- The horizontal position to show the tip (demand style only).
- The vertical position to show the tip (demand style only).

This code, used in the Init of a textbox, displays the standard style, centered, two-line tooltip shown in **Figure 7**:

```
Thisform.ctlTip.RegTool(This, .F., 1, 'Over textbox 1' + CHR(10) + 'Anything else?')
```

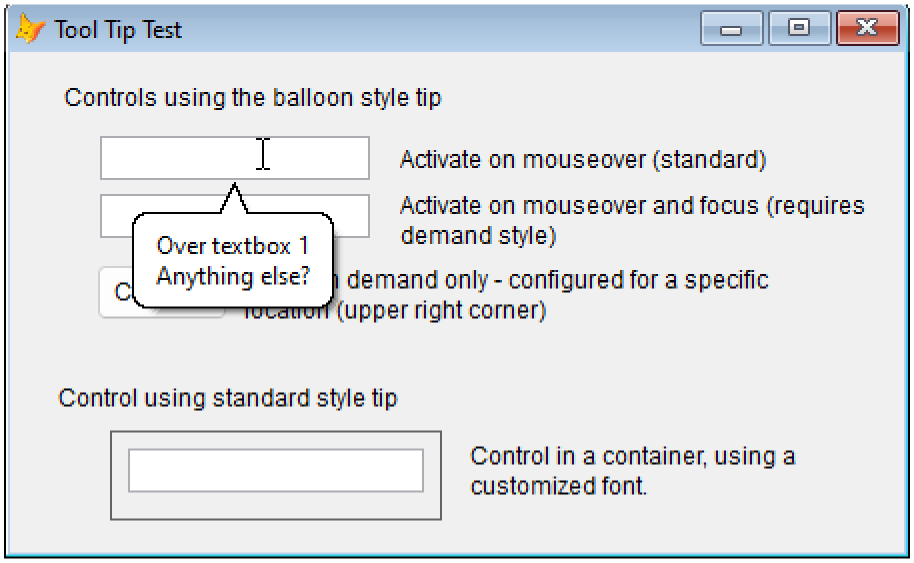

Tool Tip Test
Controls using the balloon style tip
Activate on mouseover (standard)
Activate on mouseover and focus (requires demand style)
Over textbox 1
Anything else?
demand only - configured for a specific location (upper right corner)
Control using standard style tip
Control in a container, using a customized font.

Figure 7. A standard style, centered, two-line tooltip.

This code displays the demand style tooltip with a large icon and title shown in **Figure 8**:

```
Thisform.ctlTip.RegTool(This, .T., 0, 'Over Textbox 2', 'Big Warning Icon', 5, 2)
```

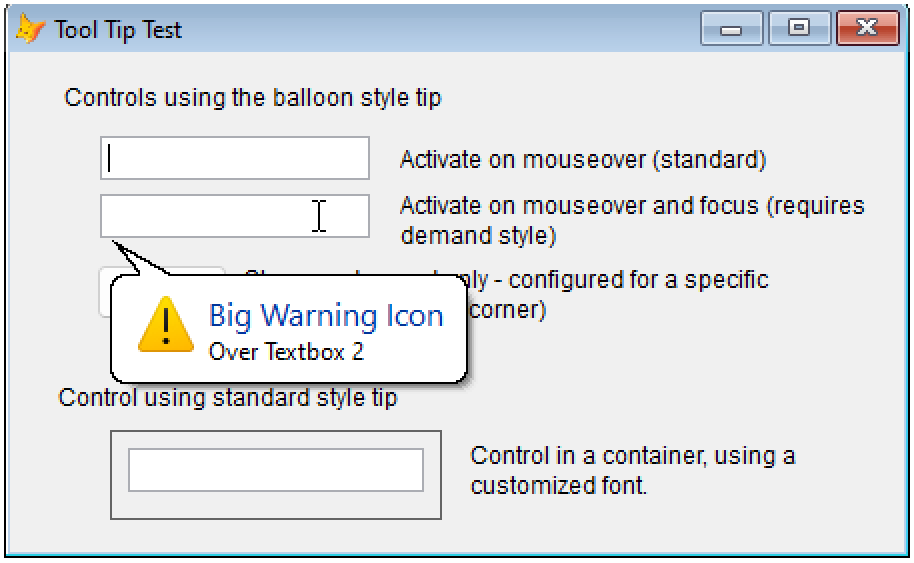
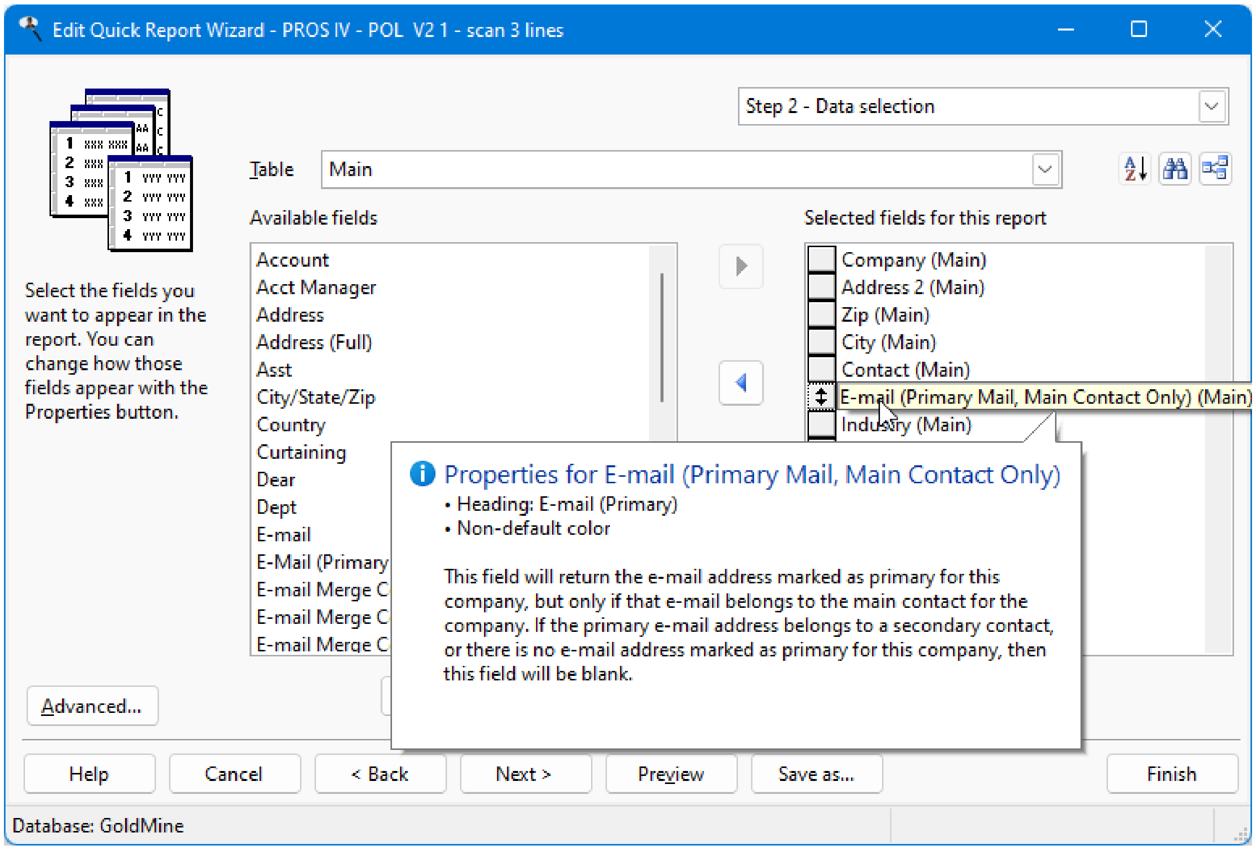

Tool Tip Test
Controls using the balloon style tip
Activate on mouseover (standard)
Activate on mouseover and focus (requires demand style)
...ly - configured for a specific ... corner)
Big Warning Icon
Over Textbox 2
Control using standard style tip
Control in a container, using a customized font.

Figure 8. A demand style tooltip with a large icon and title.

I really like balloon-style tooltips because they can display a lot more information than a simple small tooltip. For example, the tooltip shown in **Figure 9** displays non-default properties and help information for the item the mouse is hovering over. If I wasn't already using the one included in Carlos Alloatti's ctl32 library, I would definitely use this control.



Select the fields you want to appear in the report. You can change how those fields appear with the Properties button.

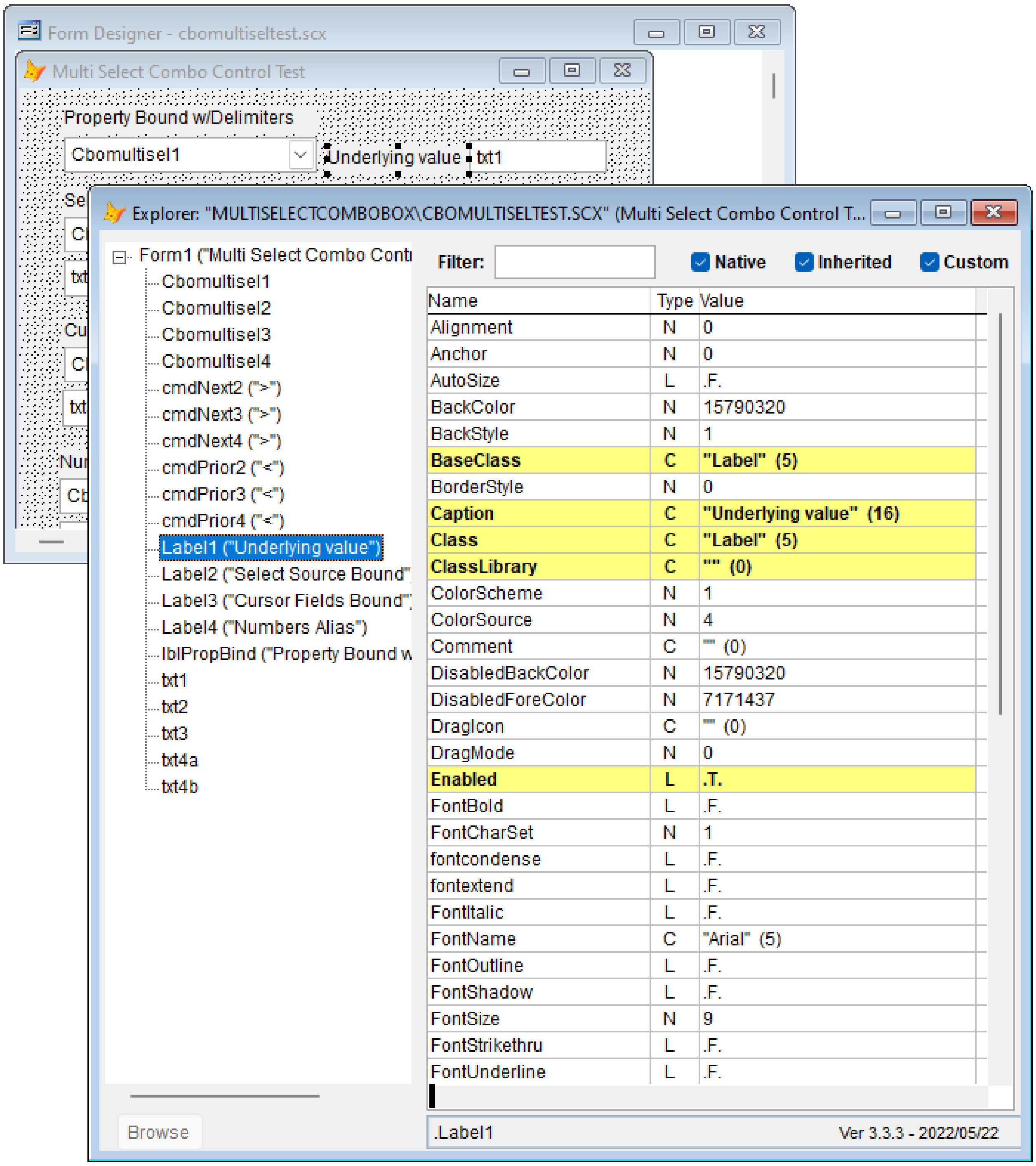
Properties for E-mail (Primary Mail, Main Contact Only)
- Heading: E-mail (Primary)
- Non-default color

This field will return the e-mail address marked as primary for this company, but only if that e-mail belongs to the main contact for the company. If the primary e-mail address belongs to a secondary contact, or there is no e-mail address marked as primary for this company, then this field will be blank.

Figure 9. Balloon-style tooltips are very useful.

Object Explorer

This project, a collaboration by Jim Nelson and the late Matt Slay, provides an explorer form (**Figure 10**) to view members of the specified object. It's similar to Tamar Granor's Object Inspector but has some additional features including filtering.


Form Designer - cbomultiseltest.scx
Multi Select Combo Control Test
Property Bound w/Delimiters
Cbomultisel1
Underlying value
txt1
Explorer: "MULTISELECTCOMBOBOX\CBOMULTISELTEST.SCX" (Multi Select Combo Control T...
Form1 ("Multi Select Combo Contr
Cbomultisel1
Cbomultisel2
Cbomultisel3
Cbomultisel4
cmdNext2 (">")
cmdNext3 (">")
cmdNext4 (">")
cmdPrior2 ("<")
cmdPrior3 ("<")
cmdPrior4 ("<")
Label1 ("Underlying value")
Label2 ("Select Source Bound"
Label3 ("Cursor Fields Bound"
Label4 ("Numbers Alias")
lblPropBind ("Property Bound w
txt1
txt2
txt3
txt4a
txt4b
Filter:
Native
Inherited
Custom
Browse
.Label1
Ver 3.3.3 - 2022/05/22

Name	Type	Value
Alignment	N	0
Anchor	N	0
AutoSize	L	.F.
BackColor	N	15790320
BackStyle	N	1
BaseClass	**C**	**"Label" (5)**
BorderStyle	N	0
Caption	**C**	**"Underlying value" (16)**
Class	**C**	**"Label" (5)**
ClassLibrary	**C**	**"" (0)**
ColorScheme	N	1
ColorSource	N	4
Comment	C	"" (0)
DisabledBackColor	N	15790320
DisabledForeColor	N	7171437
DragIcon	C	"" (0)
DragMode	N	0
Enabled	**L**	**.T.**
FontBold	L	.F.
FontCharSet	N	1
fontcondense	L	.F.
fontextend	L	.F.
FontItalic	L	.F.
FontName	C	"Arial" (5)
FontOutline	L	.F.
FontShadow	L	.F.
FontSize	N	9
FontStrikethru	L	.F.
FontUnderline	L	.F.

Figure 10. Object Explorer shows the members of the specified control and other controls on the form.

To display Object Explorer, `DO FORM Explorer WITH loObject`, where loObject is a reference to the selected object. One useful tip is to use SYS(1270), a reference to the object under the mouse, for loObject. Another tip is to configure a Thor hotkey (Thor is another VFPX project) to invoke Object Explorer; I configured Thor so pressing F9 displays Object Explorer.

You might be wondering what use Object Explorer is when the Property Window shows essentially the same information. While it can be called from the Form and Class Designers, it's really intended to be called from a running form, whether in the VFP IDE or even a runtime EXE. Not only does it display the current values of properties, which can change dynamically at runtime, you can also change the values of properties by double-clicking the current value and entering a new one. To include Object Explorer in your application so it can be used at runtime, add Explorer.vcx and Explorer.scx to your project. I've started doing that with my applications so pressing a hotkey allows me to see and even modify the current state of the controls in a form.

OOP Reports

This project, which is one of mine, provides an object-oriented wrapper to FRX reports. This allows you to programmatically create or modify reports at runtime. For example, Craig Boyd's GridExtras, which many VFP developers use to add features such as column selection and Microsoft Excel output to grid controls, uses the project to create and preview a report from the grid on demand.

OOP Reports consists of numerous classes, representing a report and the various bands and controls that make up a report. All classes are contained in SFRepObj.vcx, so you simply need to include that in your application. To create a report programmatically, instantiate an SFReportFile object, add bands, fields, labels, and other controls to it by calling various Add methods, set properties of those objects, and then call Save to generate the FRX. **Listing 10** shows the code used to create the report shown in **Figure 11**. Sample file accompanying the project also show how to modify an existing report programmatically.

Listing 10. OOP Reports allows you to create a report programmatically.

```
loReport = newobject('SFReportFile', 'SFRepObj.vcx')
loReport.cReportFile  = lcFolder + 'EmployeeReport.frx'
loReport.lSummaryBand = .T.
loReport.cFontName    = 'Arial'
loReport.cUnits       = 'Inches'
loReport.nLeftMargin  = 0.5

* Set the height of the page header band.

loPageHeader = loReport.GetReportBand('Page Header')
loPageHeader.nHeight = 1

* Set the height of the detail band.
```

```
loDetail = loReport.GetReportBand('Detail')
loDetail.nHeight = 3

* Set the height of the summary band.

loSummary = loReport.GetReportBand('Summary')
loSummary.nHeight = 0.5

* Insert column heading and field records into the appropriate band.

lnFields = afields(laFields)
for lnI = lnFields to 1 step -1
   if not inlist(laFields[lnI, 1], 'FIRST_NAME', 'LAST_NAME', 'PHOTO')
      adel(laFields, lnI)
      dimension laFields[alen(laFields, 1) - 1, alen(laFields, 2)]
   endif not inlist(laFields[lnI, 1], ...
next lnI
lnFields = alen(laFields, 1)
lnVPos   = 0
lnWidth  = 0
lnSeparation = 0.02
for lnI = 1 to lnFields

* Get the attributes for the current column.

   lcField  = laFields[lnI, 1]
   lcColumn = proper(strtran(lcField, '_', ' '))
   lcColumn = lcColumn + ':'

* Insert the field heading.

   loObject = loDetail.Add('Text')
   loObject.cExpression = lcColumn
   loObject.nVPosition  = lnVPos
   loObject.nHPosition  = 0
   loObject.lFontBold   = .T.

* Insert the field.

   if laFields[lnI, 2] = 'G'
      loObject = loDetail.Add('Image')
      loObject.cImageSource = lcField
      loObject.nImageSource = 1
      loObject.nWidth       = 2
      loObject.nHeight      = 2
      loObject.nVPosition   = lnVPos + 0.05
      loObject.nHPosition   = 1.05
      loObject = loDetail.Add('Box')
      loObject.nWidth     = 2.1
      loObject.nHeight    = 2.1
      loObject.nVPosition = lnVPos
      loObject.nHPosition = 1
   else
      loObject = loDetail.Add('Field')
      loObject.cExpression = lcField
      loObject.nWidth      = 5
      loObject.nVPosition  = lnVPos
      loObject.nHPosition  = 1
   endif laFields[lnI, 2] = 'G'

* Update the current vertical position and report width.

   lnWidth = max(lnWidth, loObject.nHPosition + loObject.nWidth)
   lnVPos  = lnVPos + loObject.nHeight + lnSeparation
next lnI

* Insert a line at the bottom of the detail band.

loObject = loDetail.Add('Line')
loObject.nWidth     = lnWidth
loObject.nVPosition = lnVPos + 0.3
loObject.nHPosition = 0

* Insert the page header.

loObject = loPageHeader.Add('Field')
loObject.cExpression = '"Employee List"'
loObject.nWidth      = lnWidth
loObject.nHPosition  = 0
loObject.nVPosition  = 0
loObject.cAlignment  = 'Center'
loObject.lFontBold   = .T.
loObject.nFontSize   = 24
loObject.nForeColor  = rgb(255, 0, 0)

* Insert the date.

loObject = loPageHeader.Add('Field')
loObject.cExpression = 'date()'
loObject.nWidth      = 1
loObject.nHPosition  = 0
loObject.nVPosition  = 0.5
loObject.lFontBold   = .T.

* Insert the page number.

lnPagenoWidth = 0.2
loObject = loPageHeader.Add('Text')
loObject.cExpression = 'Page'
loObject.nVPosition  = 0.5
loObject.nHPosition  = lnWidth - lnPagenoWidth - 0.4
loObject.lFontBold   = .T.
loObject = loPageHeader.Add('Field')
loObject.cExpression = '_pageno'
loObject.nWidth      = lnPagenoWidth
loObject.nVPosition  = 0.5
loObject.nHPosition  = lnWidth - lnPagenoWidth
loObject.cAlignment  = 'Right'
loObject.lFontBold   = .T.
```

```
* Insert a line.

loObject = loPageHeader.Add('Line')
loObject.nWidth      = lnWidth
loObject.nVPosition  = 0.75
loObject.nHPosition  = 0
loObject.nPenSize    = 6

* Add a total in the summary band.

loObject = loSummary.Add('Field')
loObject.cExpression = 'ltrim(str(lnCount)) + " record" + ' + ;
   'iif(lnCount = 1, "", "s") + " printed"'
loObject.nWidth      = 2
loObject.nVPosition  = 0.1
loObject.nHPosition  = 0
loObject.lFontBold   = .T.

* Create the report variables.

loVariable = loReport.CreateVariable()
loVariable.cName         = 'lnCount'
loVariable.cValue        = 1
loVariable.cInitialValue = 0
loVariable.cTotalType    = 'Sum'

* Set the height of the page footer band.

loPageFooter = loReport.GetReportBand('Page Footer')
loPageFooter.nHeight = 0

* Create the report, then let's run it.

loReport.Save()
modify report (loReport.cReportFile)
report form (loReport.cReportFile) preview

* Specifically release the report object so proper object cleanup occurs.

loReport.Release()
```

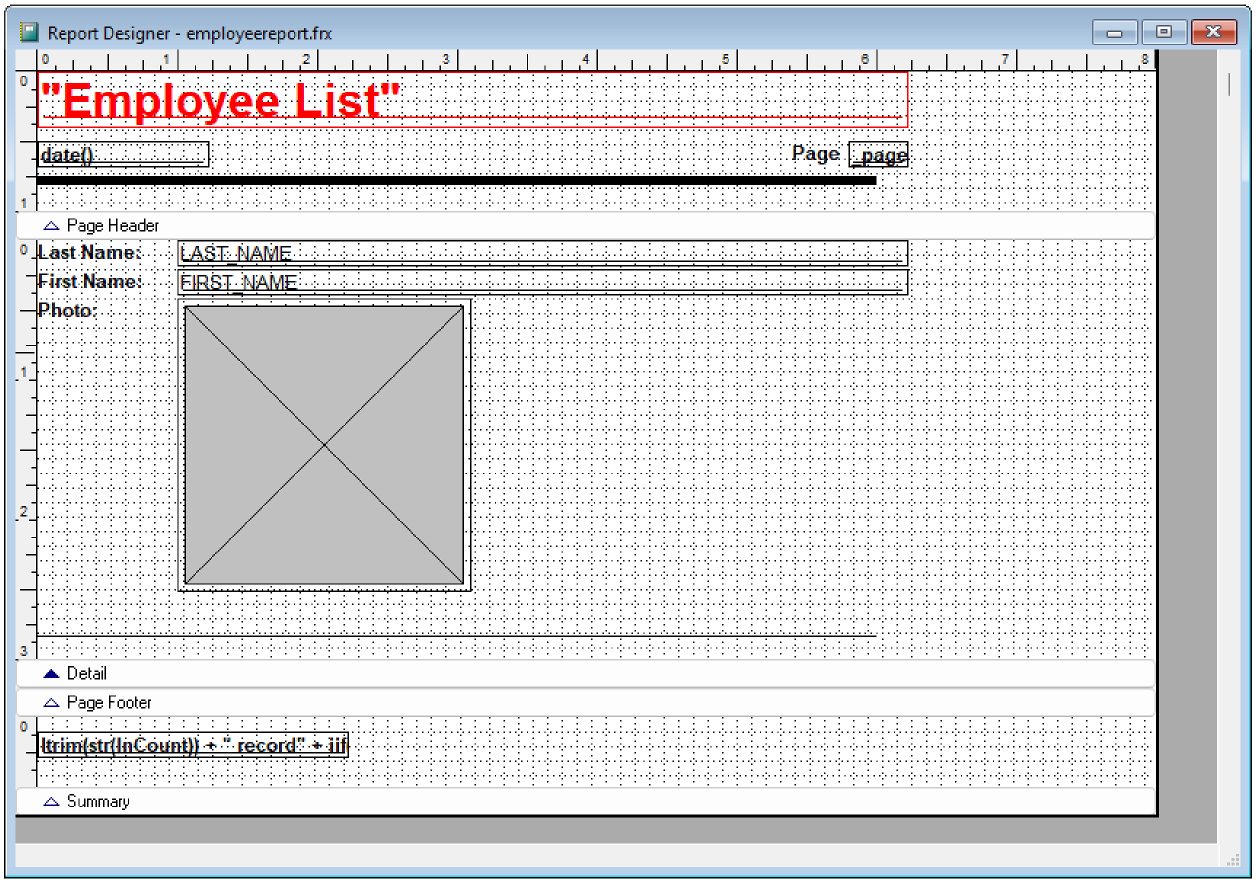


Figure 11. This report was created programmatically.

OOP Reports is the foundation for my company's main product, Stonefield Query, which allows users to create and run reports using a nice wizard interface without having to work in the Report Designer unless they want to.

SFMail

Sending emails from applications is becoming harder and harder. For a long time, most mail servers used Basic Authentication, which is simply providing a username and password to connect to the server. However, more and more mail services are moving away from Basic Authentication and implementing Modern Authentication, also known as OAuth2. Modern Authentication uses a two-step process to connect to a mail server: first obtaining a token (a string) from a web server, then using that token to connect to the mail server. As of May 2022, Google has stopped supporting Basic Authentication and in October 2022, Microsoft 365 (including Microsoft Exchange Server) is following suit.

There are a lot of ways to send emails from VFP applications using Basic Authentication: using CDO, West Wind's libraries, Chilkat, and so on. However, as far as I know, only Chilkat supports Modern Authentication.

SFMail is another of my projects. It's a library to send emails from VFP applications. It supports the following:

- Text or HTML body
- Attachments
- Normal, CC, and BCC recipients
- SMTP or MAPI
- Basic or Modern Authentication
- Adjustable timeout
- Adjustable security settings
- Diagnostic logging

Unlike some other VFPX projects, SFMail consists of several files you must distribute with your application:

- BouncyCastle.Crypto.dll
- CLRHost.dll
- MailKit.dll
- MIMEKit.dll
- SMTPLibrary2.dll
- VFPExMAPI.fll (only needed if you use MAPI)
- wwDotNetBridge.dll

Add SFMail.prg and wwDotNetBridge.prg to your project. When you want to send an email, instantiate an SFMail object, set its properties, and call the SendMail method. **Listing 11** is an example using Modern Authentication. Set the cOAuth* properties to use Modern Authentication or leave them blank to use Basic Authentication. For obvious reasons, this example reads sensitive information from an INI file I haven't included with the sample files accompanying this document. In a real application, you'd likely do something similar but store the values encrypted and decrypt them when setting the SFMail properties.

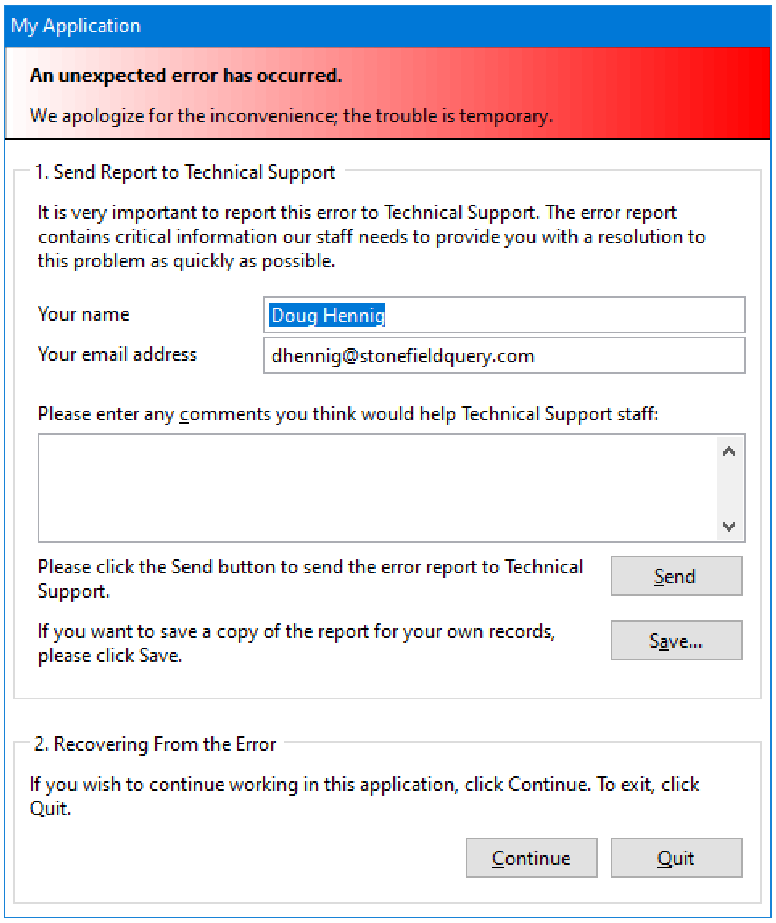
Listing 11. SFMail can send emails using Modern Authentication.

```
set path to SFMail
loMail = newobject('SFMail', 'SFMail.prg')
with loMail
   .cSubject           = 'Test email'
   .cBody              = 'This is a test message. ' + ;
      '<strong>This is bold text</strong>. ' + ;
      '<font color="red">This is red text</font>'
   .cUser              = ReadINI('DontDeploy.ini', 'Email', 'User')
   .cPassword          = ReadINI('DontDeploy.ini', 'Email', 'Password')
   .cServer            = 'smtp.office365.com'
   .nSMTPPort          = 587
   .cSenderEmail       = .cUser
   .cSenderName        = 'Doug Hennig'
   .cRecipients        = 'doug@doughennig.com'
   .cOAuthURL          = ReadINI('DontDeploy.ini', 'Email', 'OAuthURL')
   .cOAuthClientID     = ReadINI('DontDeploy.ini', 'Email', 'OAuthClientID')
   .cOAuthScope        = 'https://outlook.office.com/SMTP.Send'
   .cOAuthClientSecret = ReadINI('DontDeploy.ini', 'Email', 'cOAuthClientSecret')
   llReturn = .SendMail()
   if llReturn
      messagebox('Message sent')
   else
      messagebox('Message not sent: ' + .cErrorMessage)
   endif llReturn
endwith
```

I use SFMail in almost every application I write because emailing is becoming almost a standard feature of most applications.

Error Handler

This project, which is another of mine, provides a highly configurable and customizable error handler for any VFP application. It supports logging error information to a table, displaying an easy-to-understand, localizable dialog to the user (**Figure 12**), notifying support staff about the error via email or support ticket (optionally including a screen shot), and recovering from the error (either continuing in the application but not returning to the method that caused the error or terminating the application).



My Application

An unexpected error has occurred.

We apologize for the inconvenience; the trouble is temporary.

1. Send Report to Technical Support

It is very important to report this error to Technical Support. The error report contains critical information our staff needs to provide you with a resolution to this problem as quickly as possible.

Your name	Doug Hennig
Your email address	dhennig@stonefieldquery.com

Please enter any comments you think would help Technical Support staff:

Please click the Send button to send the error report to Technical Support.

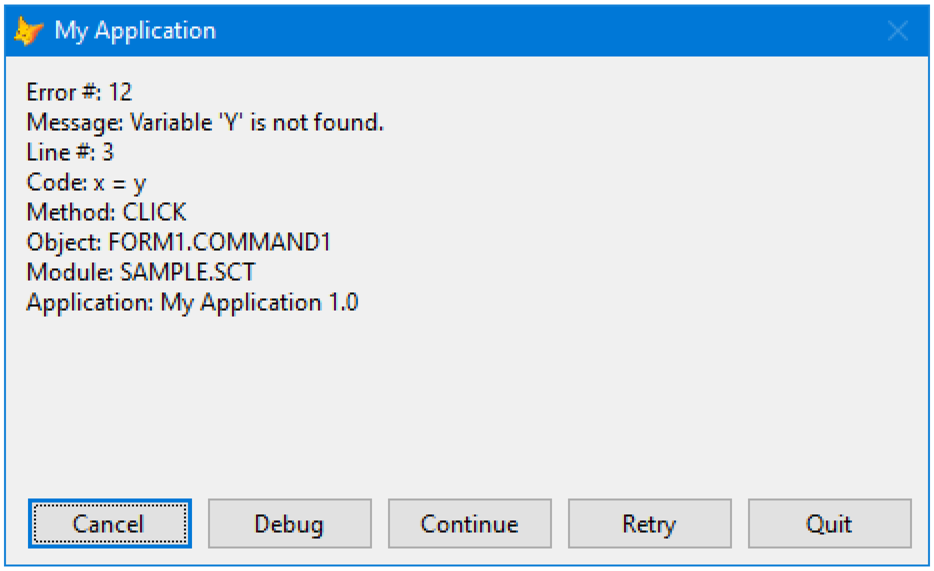
If you want to save a copy of the report for your own records, please click Save.

2. Recovering From the Error

If you wish to continue working in this application, click Continue. To exit, click Quit.

Figure 12. The error dialog seen by end-users.

While this dialog is useful for end-users, it isn't for developers because it doesn't provide a way to debug the problem. So, when an error occurs, developers get the dialog shown in **Figure 13** instead.



Error #: 12
Message: Variable 'Y' is not found.
Line #: 3
Code: x = y
Method: CLICK
Object: FORM1.COMMAND1
Module: SAMPLE.SCT
Application: My Application 1.0

Figure 13. The error dialog displayed for developers gives options a developer needs to help debug the problem.

Error Handler has very detailed documentation, including how to configure it exactly as you desire. The main class is SFErrorMgr in SFErrorMgr.vcx, but there are some supporting classes and class libraries as well.

Listing 12 shows how to set up an error handler for an application.

Listing 12. Instantiate SFErrorMgr and set some properties to add an error handler to your application.

```
oError = newobject('SFErrorMgr', 'SFErrorMgr.vcx', '', 'My Application', .T., ;
    'oError')
with oError

* If we're in "debug" mode, display a developer's dialog when an error occurs.

    if llDebugMode
        .lShowDebug      = .T.
        .cMessageClass   = 'SFErrorMessage'
        .cMessageLibrary = 'SFErrorMgr.vcx'
    endif llDebugMode

* Set properties to customize the behavior.

    .cAppName      = 'My Application'
    .cVersion      = '1.0'
    .cErrorLogFile = 'Errorlog.dbf'
    .lScreenShot   = .T.
        && take a screen shot when an error occurs

* Set the current user's name and email address.

    .cContact = 'Doug Hennig'
    .cEmail   = 'doug@doughennig.com'
```

```
* Set the email settings.

   .cRecipient   = 'dhennig@stonefieldquery.com'
   .cSenderEmail = 'dhennig@stonefieldquery.com'
   .cMailServer  = 'smtp.office365.com'
   .nSMTPPort    = 587
   .cUserName    = ReadINI('DontDeploy.ini', 'Email', 'User')
   .cPassword    = Encrypt(ReadINI('DontDeploy.ini', 'Email', 'Password'),
.cUserName)

* Set the localizer settings.

   .cLanguage      = 'English'
   .cResourceTable = 'Resource.dbf'
endwith
```

The Init of SFErrorMgr sets the global error handler to:

```
oError.ErrorHandler(error(), sys(16), lineno(), alias(), set('DATASESSION'))
```

so any error not trapped by TRY or the Error method of an object calls the ErrorHandler method of the SFErrorMgr object.

To see it in action, edit ErrorExample.prg and set the cRecipient, cMailServer, nSMTPPort, cUserName, and cPassword settings to the appropriate values for your email server. Then run that program and click the Cause an Error button in the form that appears. This causes an error by executing X = Y without defining a variable named Y. Click the Send button in the error dialog that appears to email an error report to the address you specified in cRecipient, click the Save button to save the error report to a text file, and click Continue to handle the error without exiting the “application.” Then click the Cause an Error Inside TRY button in the sample form and notice the error dialog no longer has a Continue button; the only option is to terminate the program. Set llDebugMode to .T. in ErrorExample.prg and run it again. This uses the developer version of the error dialog.

Deploy these files along with your application’s executable:

- BouncyCastle.Crypto.dll
- ClrHost.dll
- InTry.dll
- MailKit.dll
- MimeKit.dll
- SMTPLibrary2.dll
- System.app (only needed if you want screen shots taken)
- VFPEncryption71.fll

- wwDotNetBridge.dll

Notice that some of these files are the same ones SFMail uses; that's because Error Handler uses SFMail to email support staff about an error.

Error Handler makes it easy to add a full-featured, configurable error handler to any application.

Ribbon

This project, which is another of mine, provides a ribbon control similar to that of Microsoft Office 365 to VFP forms. **Figure 14** shows the sample form that comes with the project.

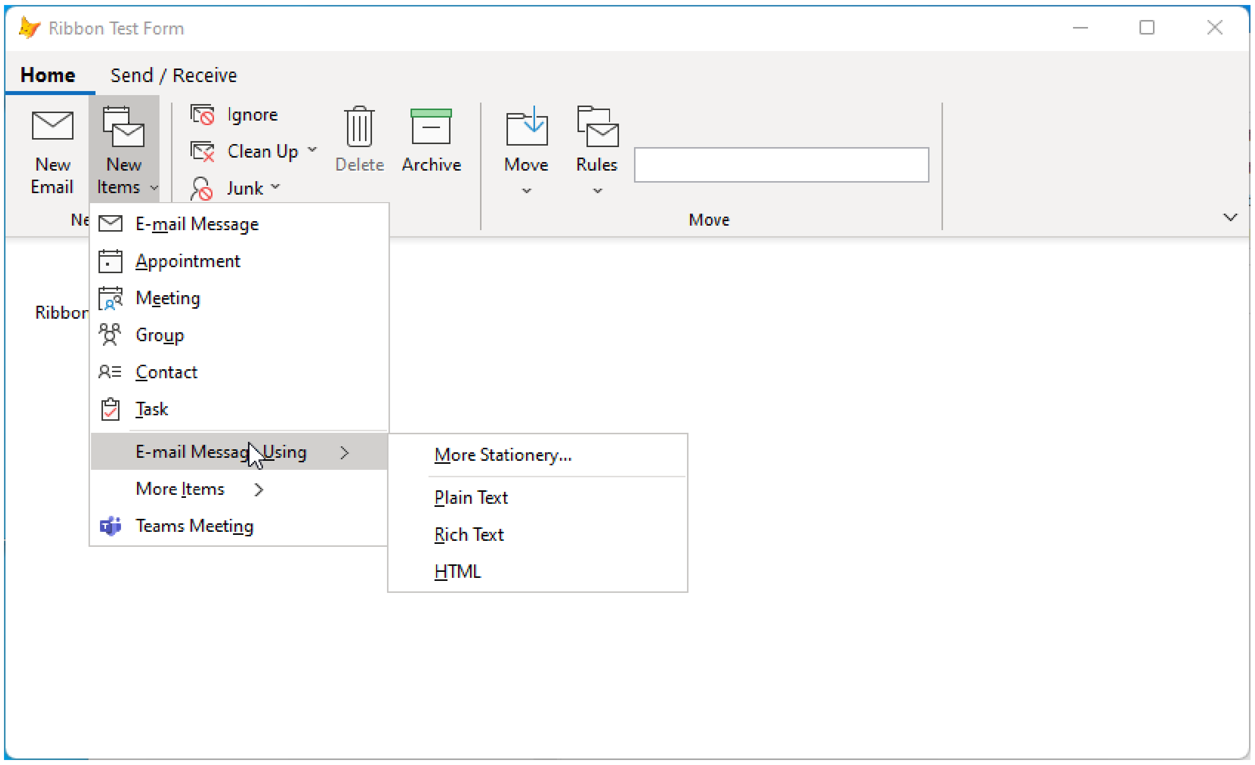


Figure 14. The Ribbon control provides a Microsoft 365-like control for VFP forms.

A ribbon consists of multiple components, shown in **Figure 15**.

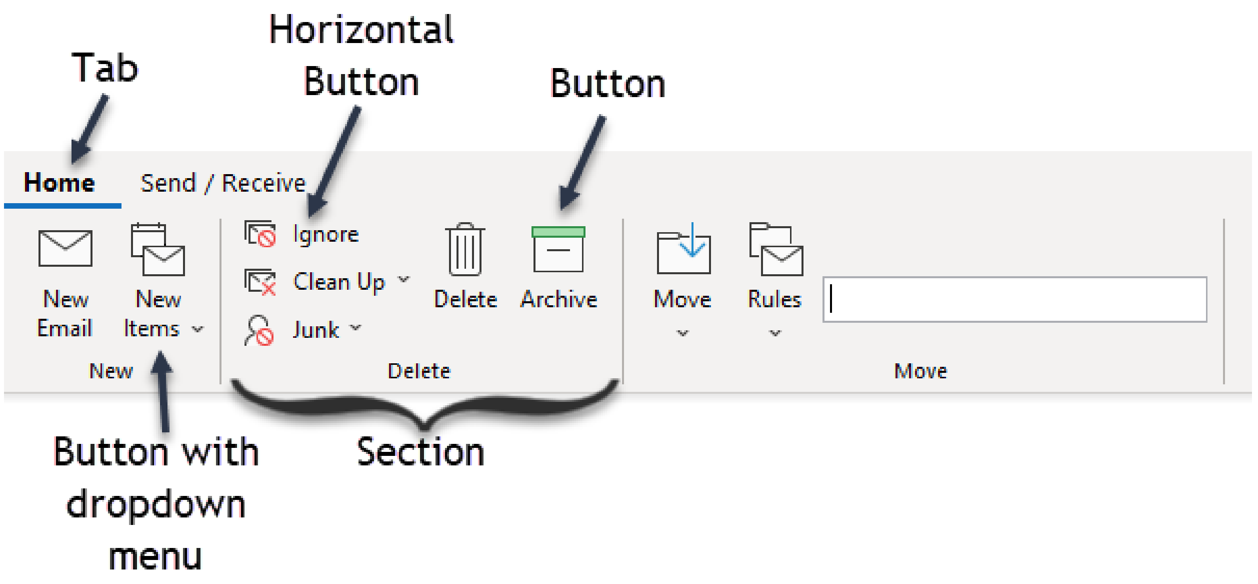


Figure 15. A ribbon consists of multiple components.

Drop the ribbon control on a form or class, then set it up programmatically by calling AddTab to add tabs to the control, AddSection to add a section to a tab, and AddButton to add a button to a section. Adding a tab, section, or button returns a reference to the new component, so set properties of the component, such as Caption and Image, to control its appearance and behavior. **Listing 13** shows some of the code from the Init method of the sample form.

There are two types of buttons: regular and horizontal. The difference is that regular buttons are laid out left to right and have a 32 x 32 image with a caption below while horizontal buttons are stacked one on top of another and have a 16 x 16 image with a caption to the right. In the Delete section of the Home tab of the sample form, the Ignore, Clean Up, and Junk buttons are horizontal buttons.

To determine what happens when a button is clicked, set its Command property to the code to execute. The code is executed via the EXECSCRIPT() function, so it can consist of multiple statements separated by carriage returns if necessary.

A button can have a dropdown menu; if it does, a small down arrow appears in the button caption. To add items to the menu, call AddBar. Several of the buttons in the sample form, including New Items and Clean Up, have a menu.

Listing 13. Setting up the ribbon involves calling AddTab, AddSection, and AddButton.

```
with This.oRibbon

* Set up the Home tab.

    loTab = .AddTab('Home')
    with loTab
        .Caption = 'Home'
```

```
* Set up the New section.

      loSection = .AddSection()
      with loSection
          .Caption = 'New'
          loButton = .AddButton()
          with loButton
              .Caption = 'New' + chr(13) + 'Email'
              .Image   = lcImagePath + 'newemail.png'
              .Command = "messagebox('New email')"
          endwith
          loButton = .AddButton('NewItems')
          with loButton
              .Caption = 'New' + chr(13) + 'Items'
              .Image   = lcImagePath + 'newitems.png'
              .AddBar('E-\<mail Message', 'Thisform.SomeMethod()', ;
                  lcImagePath + 'newemailsmall.png')
              .AddBar('\<Appointment', "messagebox('Appointment')", ;
                  lcImagePath + 'appointmentsmall.png')
              .AddBar('M\<eeting', "messagebox('Meeting')", ;
                  lcImagePath + 'meetingsmall.png')
              .AddBar('Gro\<up', "messagebox('Group')", ;
                  lcImagePath + 'groupsmall.png')
              .AddBar('\<Contact', "messagebox('Contact')", ;
                  lcImagePath + 'contactsmall.png')
              .AddBar('\<Task', "messagebox('Task')", ;
                  lcImagePath + 'tasksmall.png')
              .AddBar()
              loBar = .AddBar('E-mail Message \<Using')
              loBar.AddBar('\<More Stationery...', "messagebox('More stationery')")
              loBar.AddBar()
              loBar.AddBar('\<Plain Text')
              loBar.AddBar('\<Rich Text')
              loBar.AddBar('\<HTML')
              loBar = .AddBar('More \<Items', "messagebox('More Items')")
              loBar.AddBar('\<Post in This Folder')
              loBar.AddBar('Contact \<Group')
              loBar.AddBar('Task Re\<quest')
              .AddBar('Teams Meeti\<ng', "messagebox('Teams Meeting')", ;
                  lcImagePath + 'teamsmeetingsmall.png')
          endwith
      endwith

* Set up the Delete section.

      loSection = .AddSection()
      with loSection
          .Caption = 'Delete'
          loButton = .AddHorizontalButton()
          with loButton
              .Caption = 'Ignore'
              .Image   = lcImagePath + 'ignore.png'
          endwith
          loButton = .AddHorizontalButton()
```

```
            with loButton
                .Caption = 'Clean Up'
                .Image   = lcImagePath + 'cleanup.png'
                .AddBar('\<Clean Up Conversation', , lcImagePath + 'cleanup.png')
                .AddBar('\<Clean Up Folder')
                .AddBar('\<Clean Up Folder & Subfolders')
            endwith
            loButton = .AddHorizontalButton()
            with loButton
                .Caption = 'Junk'
                .Image   = lcImagePath + 'junk.png'
                .AddBar('\<Block Sender', , lcImagePath + 'junk.png')
                .AddBar('Never Block \<Sender')
                .AddBar("Never Block Sender's \<Domain (@example.com)")
                .AddBar('Never Block this Group or \<Mailing List')
                .AddBar()
                .AddBar('\<Not Junk', , lcImagePath + 'newemailsmall.png', '.F.')
                    && .F. is passed as a string because it's an expression
                    && that's evaluated
                .AddBar('Junk E-mail \<Options...')
            endwith
            loButton = .AddButton()
            with loButton
                .Caption           = 'Delete'
                .Image             = lcImagePath + 'delete.png'
                .EnabledExpression = '.F.'
            endwith
            loButton = .AddButton()
            with loButton
                .Caption = 'Archive'
                .Image   = lcImagePath + 'archive.png'
            endwith
        endwith
    endwith

* Auto-select the Home tab.

    .Home.Selected = .T.
endwith
```

You can add other types of controls besides buttons to a section, such as the textbox in the Move section of Figure 14. To do that, call the AddControl method of the section, passing the name, class, and library for the control.

As you can see in **Figure 16**, the ribbon can be collapsed if you want more screen real estate. To enable that, set the AllowShowTabsOnly property of the ribbon to .T. to display an arrow at the far right of the ribbon. When the user clicks that arrow, a menu displays with Show Tabs Only and Always Show Ribbon items. To control what happens when the user chooses Show Tabs Only, bind the OnShowTabsOnly event of the ribbon control to a method of the form, and in that method, adjust the positions of the controls below the ribbon up or down, depending on whether the tlShowTabsOnly parameter is .T. or .F.

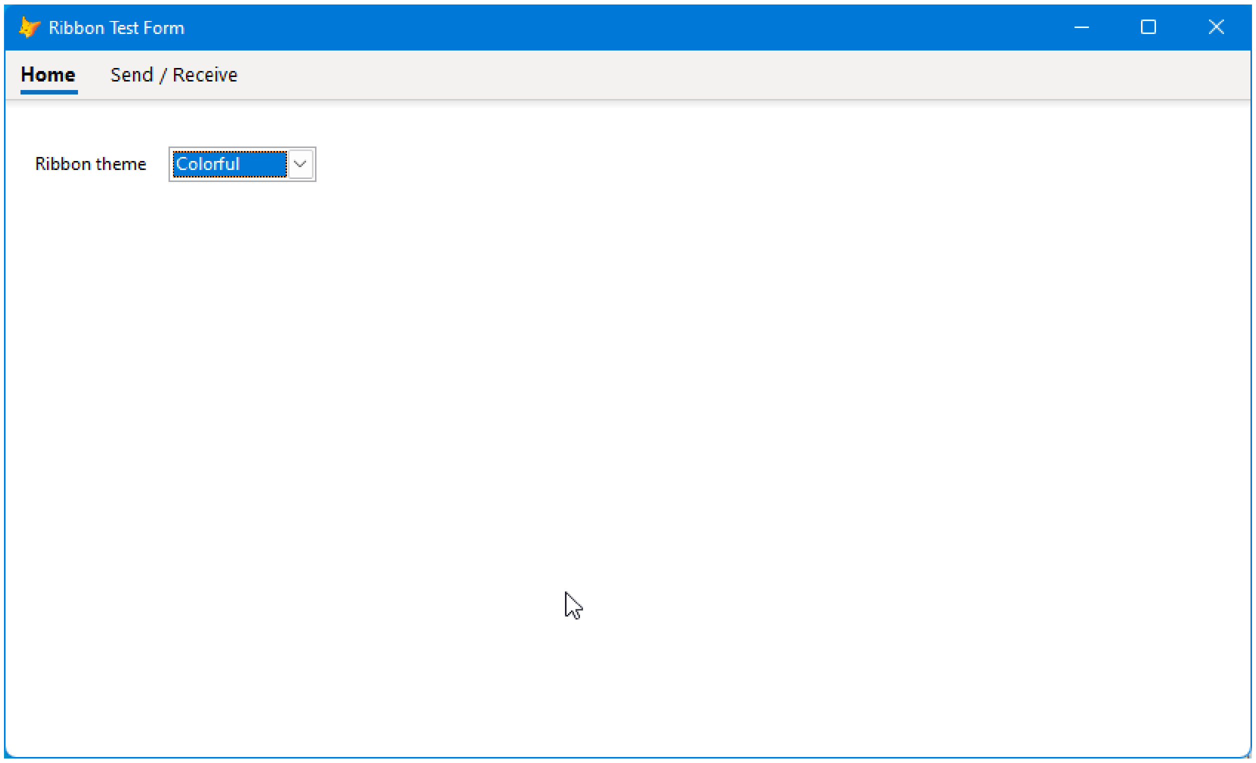
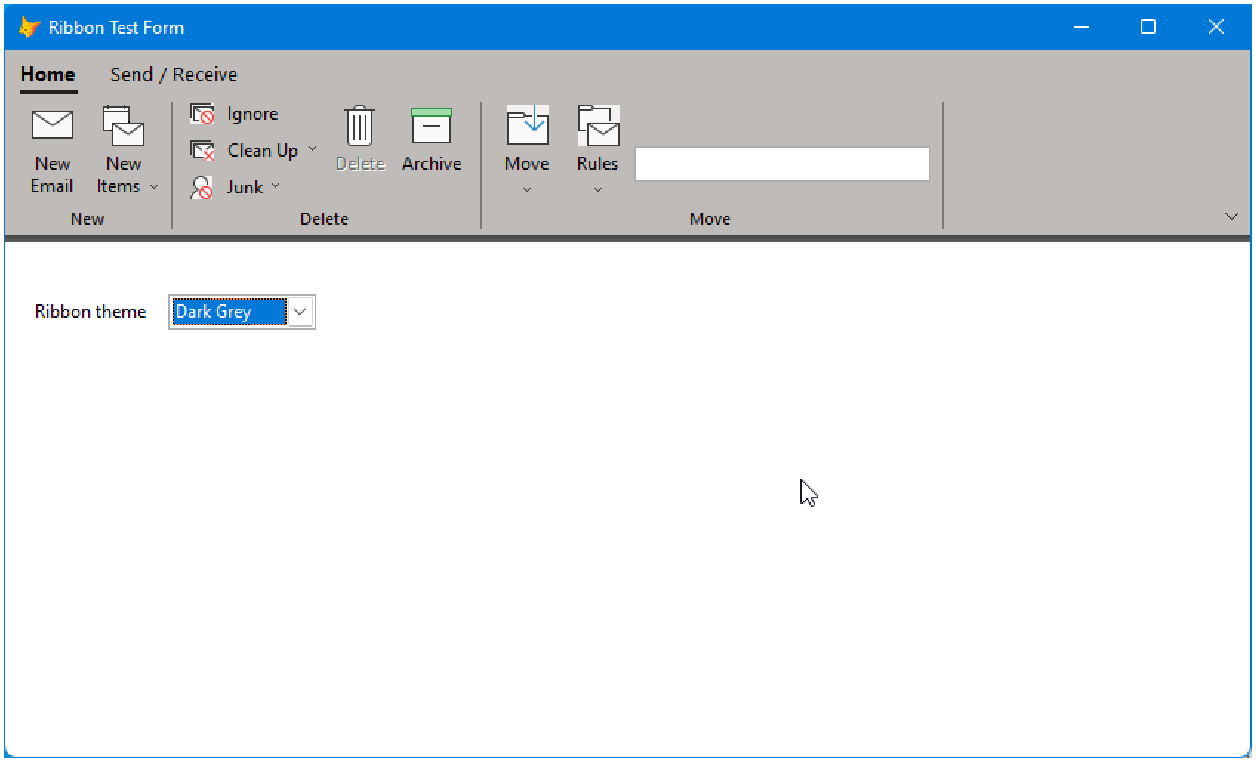


Figure 16. You can hide the ribbon for more screen real estate.

The ribbon control supports themes. Currently, two themes are available: Colorful, shown in the previous images, and Dark Grey, shown in **Figure 17**. To change the theme, set the Theme property. For example, this code adds a shortcut menu to the ribbon itself with items to select the desired theme:

```
.AddBar('Change Theme to Colorful', ;
   "Thisform.oRibbon.Theme = 'Colorful'" + chr(13) + "Thisform.Refresh()", , ;
   "Thisform.oRibbon.Theme <> 'Colorful'")
.AddBar('Change Theme to Dark Grey', ;
   "Thisform.oRibbon.Theme = 'Dark Grey'" + chr(13) + "Thisform.Refresh()", , ;
   "Thisform.oRibbon.Theme <> 'Dark Grey'")
```



Ribbon theme: Dark Grey

Figure 17. The ribbon includes the Dark Grey theme.

Themes are defined in RibbonThemes.xml, so additional themes can be added.

To deploy the ribbon with your application, add the following files to your project:

- SFRibbon.vcx
- SFGDIMeasureString.prg
- SFRibbonDown.png
- SFRibbonRight.png
- SFRibbonCheck.png
- SFRibbonDownLarge.png
- RibbonThemes.xml

Also, include System.app, the GDIPlusX component that many VFPX project use, with the files installed with your application.

Not everyone likes the ribbon control, so I allow my users to switch between the traditional menu and toolbar and the ribbon on a user-by-user basis. Adding a ribbon control to your applications is one way to modernize the user interface.

Resources

The Technical Papers page of my personal web site has articles doing deep dives on several VFPX projects, including Themed Controls, Dynamic Forms, and FoxCharts. In addition, my "VFPX 2018 Edition" white paper is the previous installment of this document.

The Articles page of Tamar Granor's web site has several in-depth articles on Thor, Object Inspector, and FastXTab.

VFPX: Open Source Treasure for the VFP Developer is a little out of date but still a great resource for many VFPX projects, especially its section on Thor.

Summary

We just barely covered the surface for the more than 20 projects added to VFPX in the past four years. I hope you find some useful projects and take the time to dig into them further.

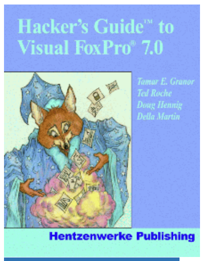
If you're new to VFPX or haven't visited in a while, check it out at http://vfpx.org and see what projects you can make use of. Of course, we welcome ideas you have for new projects as well.

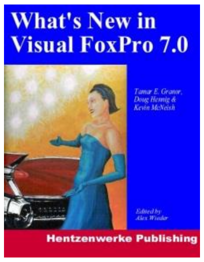
Biography

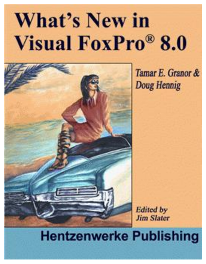
Doug Hennig is a partner with Stonefield Software Inc. He is the author of the award-winning Stonefield Query; the award-winning Stonefield Database Toolkit (SDT) (now open source); the MemberData Editor, Anchor Editor, and CursorAdapter and DataEnvironment builders that come with Microsoft Visual FoxPro; and the My namespace and updated Upsizing Wizard in Sedna. He also created several VFPX projects, including Project Explorer, OOP Menu, OOP Reports, and SFMail.

Doug is co-author of *VFPX: Open Source Treasure for the VFP Developer*, *Making Sense of Sedna and SP2*, *Visual FoxPro Best Practices For The Next Ten Years*, the *What's New in Visual FoxPro* series, and *Hacker's Guide to Visual FoxPro 7.0* (now open source). He was the technical editor of *Hacker's Guide to Visual FoxPro 6.0* and *The Fundamentals*. Doug wrote hundreds of articles in 20 years for *FoxRockX*, FoxTalk, FoxPro Advisor, Advisor Guide to Visual FoxPro, and CoDe magazines.

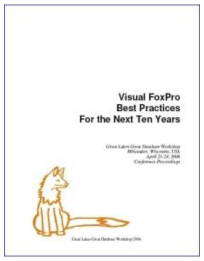
Doug spoke at every Microsoft FoxPro Developers Conference (DevCon) starting in 1997 and at user groups and developer conferences all over the world. He is one of the organizers of the Southwest Fox and Virtual Fox Fest conferences. He is one of the administrators for the VFPX VFP community extensions Web site. He was a Microsoft Most Valuable Professional (MVP) from 1996 through 2011. Doug was awarded the 2006 FoxPro Community Lifetime Achievement Award.

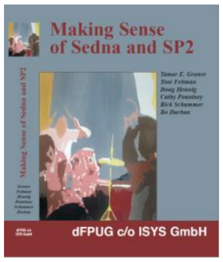

Hacker's Guide™ to Visual FoxPro® 7.0
Tamar E. Granor
Ted Roche
Doug Hennig
Della Martin
Hentzenwerke Publishing

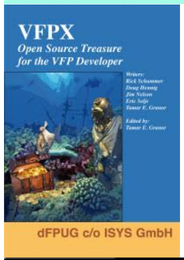

What's New in Visual FoxPro 7.0
Tamar E. Granor, Doug Hennig & Kevin McNeish
Edited by Alex Wieder
Hentzenwerke Publishing


What's New in Visual FoxPro® 8.0
Tamar E. Granor & Doug Hennig
Edited by Jim Slater
Hentzenwerke Publishing

What's New in Nine: Visual FoxPro's Latest Hits
Hentzenwerke Publishing


Visual FoxPro
Best Practices
For the Next Ten Years


Making Sense of Sedna and SP2
dFPUG c/o ISYS GmbH


VFPX
Open Source Treasure for the VFP Developer
dFPUG c/o ISYS GmbH