Developer Tools: Build, Grab, or Buy

Session Number SCH137

Richard A. Schummer
President
White Light Computing, Inc.
42759 Flis Dr.
Sterling Heights, MI 48314
Voice: 586.254.2530
Fax: 586.254.2539
E-mails: raschummer@whitelightcomputing.com
rick@rickschummer.com
Web sites: www.whitelightcomputing.com
www.rickschummer.com

Overview

Are you in the market to increase your productivity in the minute-to-minute working with Visual FoxPro? Are your clients demanding shorter deadlines and expressing the need to get their projects to market faster and faster? Are you finding certain tasks in the VFP IDE to be tedious, repetitive, or just plain old mundane and wish there was a better way? This session will cover ways of eliminating some of those tedious, repetitive tasks. This session will discuss some of the reasons to write your own tools, and then demonstrate various techniques to generate home grown developer tools. Next, this session will show some of the nicer freebies available publicly and finally discuss what third-party tools you might want to purchase to advance developer productivity.

Be Smart! (My Disclaimer)

Before we start it is very important to note that anything done with any metadata tables must be done with great care. If you compromise a metadata file you could lose minutes to years worth of work. It goes without saying, make good backups of all the metadata files and programs, after all, this is the source code of the applications and your lifeline to cash flow. I will also note that most people that do make backups (which is a small percentage of all computer users to start with), do not verify backups. Take the extra time needed for this extra protection.

Microsoft will not support issues that are raised during this session. The tables are only supported to the extent that Microsoft wrote them and places data into them. They have however made the metadata tables an open standard (through documentation) so third-party developers can utilize them and extend them for the benefit of Visual FoxPro developers.

Types of VFP Tools

There are probably as many reasons to develop developer tools as there are tools developed. I have categorized the types of developer tools into three groups.

VFP Source/Metadata Manipulators

Most of the different source code types in VFP are stored in metadata (defined in the next section). There are tools that manipulate the source code (documenting, searching for strings, printing, evaluating, etc) to help developers be more productive.

Visual FoxPro IDE Enhancements

The Visual FoxPro IDE has been enhanced in each release since 1995, yet there remain numerous shortcomings (Menu Builder comes immediately to mind) and things that are just more difficult than they need to be (four buttons to click and dialogs to respond to build an application, changing fonts in the Report Designer). The Fox team at Microsoft has been doing a bang-up job with the improvements in VFP 6 and VFP 7 to ease our pain for sure. They have also put in hooks for add-ons (like the Class Browser, Object Browser, and Coverage Profiler) and all new functionality with an extendible IDE.

Outside Resources

This category is a catch all for things like help development tools, database design tools, bug tracking tools, table and memo file recovery tools, frameworks, and icon and image editors. They are not directly linked to VFP, yet they are a necessity in building custom and vertical market applications.

Why Build?

Increased productivity has to be the number one reason to build, grab or buy a developer tool. Why do we turn to tools in our development? So we can accomplish a task faster, better, and/or more easily. There is pressure from our customers, from our bosses, and from our peers to do our jobs with more accuracy and speed than ever before. Developer tools, if they are architected and used correctly can enhance the way we develop applications. An example of this is to create a

developer menu that shortcuts a list of tools you use every day. Even something as simple as changing a directory (instead of typing CD with a 30 character directory hierarchy), or generating a build dialog that avoids the 4 steps to create an EXE. Creating shortcuts for tasks that take time can save hours on a single project.

I do not work on an auto assembly line because of one reason, redundancy (thank goodness there are folks that do enjoy this important job). I like the concept of repeatable processes, but not the idea of repeating a process hundreds of times in a row. Many developer tools are created to reduce redundancy in working with VFP. An example of this is to write a little program that loops through objects in a report to change the font from “Arial” to “Tahoma”. Changing the fonts in a report is not a difficult task, but when you have fifty reports to change in an application, this can be a very tedious task.

I have used the opportunity of building a developer tool as a mechanism to help myself and other developers learn more about VFP. There are a number of commands in VFP that I have read about in the help file and scratched my head wondering how the heck I would ever used it. Later, in developing a tool I see the use for the command. There are a number of commands and functions in Visual FoxPro that were added specifically for development of tools like builders and the custom designers.

Sometimes there is nothing available and you are between a rock and a hard place. The process you are trying to automate might just be unique to your environment or it could be that no other developer has thought up the idea before. I try to look for tools already developed and publicly available. Sometimes you are lucky and find someone else has already expended the effort and created something useful. Sometimes you find something that works, but only does a portion of what you need it to do. Sometimes you just need to crack open the hood of VFP and construct the solution on your own.

Visual FoxPro is a very mature development tool. What I mean by this is that the product has been through many updates which has given the Fox Team at Microsoft time to iron out the worst bugs and add usability features that makes development simpler or faster. Still, there are plenty of opportunities to supplement VFP features to improve a process or simplify a task.

The great thing about VFP is that it is so darn flexible.

What is Metadata?

First we need to discuss metadata. The reason for this is that much of the source code stored by VFP is stored in metadata files.

Metadata is simply “data about data”. We all build applications to store inventory, employee, and customer and other business related records on a daily basis. This is valuable data to our customers. It has meaning to the individual, business, or organization that inputs and maintains it. Metadata is information (data) that is about this “real” data. Metadata is data that describes data.

The best example of metadata is a data dictionary. Typical data dictionaries are stored in a table format. The VFP Database Container is an example of a table based data dictionary. In the data dictionary “table” is a record for each database table, view, and connection. There are records for each field of each table and view within the database. Each of these field records contains

information like field length, long field names, input masks, and comments. There is information on primary keys, candidate keys, and regular indexes as well.

So what is the big deal about metadata? In my personal opinion it gives Visual FoxPro developers a huge advantage over developers building solutions in other development tools. This session will show several examples of how we can use metadata to make our development lives easier by taking advantage of the metadata.

Different Visual FoxPro Metadata Files

Each of the Visual FoxPro power tools has an associated metadata table (except for the Query Designer). New versions of Visual FoxPro have triggered new file layouts for these tables (the most recent is the menu change in VFP 7). I am not going to document each of these changes since the Microsoft team has done a fairly good job of this already (see section on *Sources of Documentation*). Instead I will discuss some of the key differences where necessary for the benefit of this discussion and concentrate on ways for developers to utilize this information.

Metacompare

Name	Type	Id	Timestamp	Outfile	Homedir	Exclude	Mainprog	Savecode	Debug	Encrypt	Nologo	Cmntstyle	Objrev	Devinfo	Symbols	Object	Ckval	Cpid
Memo	H		724270071	Memo	Memo			T	T	F	F	1	260	Memo	memo	Memo		
Memo	R	656837271	724221289	memo	memo	T								memo	Memo	memo	27581	1252
Memo	V	661542029	724266645	memo	memo	F	F						544	memo	Memo	memo	7377	1252
Memo	P	688087897	710315720	memo	memo	F	F			F			544	memo	Memo	Memo	65417	1252
Memo	T	688092457	724221228	memo	memo	F								memo	Memo	memo	30213	1252
Memo	T	689480487	714628886	memo	memo	T								memo	Memo	memo	43673	1252
Memo	T	689480528	714233938	memo	memo	F								memo	Memo	memo	1126	1252
Memo	P	689601070	714233602	memo	memo	F	T			F			544	memo	Memo	Memo	47013	1252
Memo	x	710484092	611820381	memo	memo	F								memo	Memo	memo	2326	1252
Memo	V	710484325	724270056	memo	memo	F	F						544	memo	Memo	memo	8124	1252
Memo	T	712867925	712868019	memo	memo	F								memo	Memo	memo	26109	1252
Memo	x	714234322	713840778	memo	memo	F								memo	Memo	memo	38743	1252
Memo	V	714234445	724221289	memo	memo	F	F						544	memo	Memo	memo	63996	1252
Memo	R	714234765	724221290	memo	memo	T								memo	Memo	memo	61537	1252
Memo	x	714235577	613941248	memo	memo	F								memo	Memo	memo	53595	0
Memo	x	714508055	714508886	memo	memo	F								memo	Memo	memo	57535	0
Memo	x	714508056	714508909	memo	memo	F								memo	Memo	memo	33230	0
Memo	x	714508057	714509063	memo	memo	F								memo	Memo	memo	15807	0
Memo	x	714508061	714507978	memo	memo	F								memo	Memo	memo	61112	0
Memo	x	714508064	714509116	memo	memo	F								memo	Memo	memo	28289	0
Memo	x	714508065	714507253	memo	memo	F								memo	Memo	memo	9300	0
Memo	x	714508728	714509275	memo	memo	F								memo	Memo	memo	7445	0
Memo	W		0	memo	memo	T								memo	memo	memo		

Figure 1. *Looking at a VFP Project file via the old BROWSE command,*

Visual FoxPro application source code is stored for forms, visual classes, reports, labels, menus, database container, and projects in metadata. Programs and queries have always been stored in flat file format and are not considered metadata. VFP interprets all the metadata files at runtime except for the menus which still uses GenMenu.prg (or third-party GenMenuX.prg) to generate menu code stored in .MPRs.

User Field

Each of the VFP metadata tables (except the menus) contains a USER memo field. This field is not used by Visual FoxPro and is available solely for developers to leverage. One idea for this field is to store documentation about the object in this field. Another idea I have for this field is to store information in the project's USER memo field to indicate the last time it was backed up via a developer tool such as one that zips up all the files in a project for storage elsewhere.

TimeStamp Field

Example: TimeStamp2Date.prg (in downloads)

The TimeStamp field is a 32 bit binary encoded field that represents the date and time that the Fox Team contrived to save space in the metadata files. This scheme was developed before there was a DateTime data type available.

The algorithm is pretty straight forward once you find the correct documentation (available on the Microsoft KnowledgeBase). Here is the core code of the TimeStamp2Date.prg (available with the downloads for this session):

```
lnYear   = ((tnTimeStamp/(2**25) + 1980))
lnMonth  = ((lnYear-INT(lnYear)    )*(2**25))/(2**21)
lnDay    = ((lnMonth-INT(lnMonth)  )*(2**21))/(2**16)

lnHour   = ((lnDay-INT(lnDay)      )*(2**16))/(2**11)
lnMinute = ((lnHour-INT(lnHour)    )*(2**11))/(2**05)
lnSecond = ((lnMinute-INT(lnMinute))*(2**05))*2       &&  Multiply by two to correct
                                                      &&  truncation problem built in
                                                      &&  to the creation algorithm
                                                      &&  (Source: Microsoft Tech Support)
```

Once you have the individual pieces you can assemble a date and time that is more readable to the average human being.

Project File (.PJX/PJT)

The project file contains one header entry for the project and one record for each FoxPro object used in the application. The big change from VFP 3.0 to 5.0 is the inclusion of some source code control information.

There are several powerful things one can do with the project file. My favorite is to list all the objects in the project to create a check list (see VFP Project Lister in Tools to Grab). This can also be part of some technical documentation for the project.

```
SELECT PADR(STRTRAN(Name,CHR(0)," "), 40) AS cName, * ;
   FROM orbit ;
   ORDER BY cName, Type ;
   INTO CURSOR query
```

Another idea is a utility that scans through the project file and builds a list of objects that are included in a .ZIP file or copied to another drive for backups or bundled to take to a client site (if they have purchased the rights to the source). One specific thing I like to do is quick searches for reusable objects that are in several projects:

```
SELECT PADR(STRTRAN(Name,CHR(0),SPACE(0)), 40) AS cName, * ;
   FROM pickaproject ;
   WHERE name IN (SELECT name ;
                  FROM Orbit ) ;
   INTO CURSOR query
```

It is important to remember that all the different objects stored in a project file are stored in the Name field in relative path. Therefore the query I just coded will only work for projects that are off a common parent directory. Here is a sample of the fields from the above query:

Name	Type	Id	TimeStamp
"..\common\classes\cbaseclass.vcx"	"V"	557736201	555921818
"..\common\classes\ccustomcontrols.vcx"	"V"	558279957	555921865
"..\common\bitmaps\close.bmp"	"x"	558279957	516030464

Name	Type	Id	TimeStamp
"..\common\bitmaps\new.bmp"	"x"	558279957	550087046
"..\..\devwtool\vfp50b344\foxpro.h"	"T"	558530861	555024384

We will see in the Project Lister that there are routines that can be leveraged to build a complete path and decode the 32-bit TimeStamp field to a date/time format we are used to reading.

Project Object and Files Object COM Interface

Example: ProjectFileObjPems.prg (in downloads)

The project object is built into Visual FoxPro and cannot be subclassed. It provides a COM interface to the project file. In the past developers had to "hack" the project file by opening it up as a VFP free table via the USE command. The use of the SQL-Select command and the project opened as a free table is a powerful combination to extract information that is stored in the metadata. While some of these techniques are still useful, the project object reduces the need to "hack" the project metadata.

There is one difference in accessing the COM interface to the project from the standard way you might be accustomed. COM objects are instantiated via the `CREATEOBJECT()`, `CREATEOBJECTEX()`, or `NEWOBJECT()` functions. The project object is created for you any time a project is opened. The project object is created for each open project and is accessed via the `_VFP` application object. This object gives you access to several properties and methods to gain information about the project. Here is some code that accesses key information from the project:

```
WITH _vfp.ActiveProject
   ? .VersionNumber   && Displays next build version number
   ? .MainFile        && Displays name (and path) of main file
   .CleanUp()         && Packs the project metadata
ENDWITH
```

The files object and files collection of the project object is how one gets access to the individual files within the project. There are a number of properties that are associated with each file. There are also behaviors that can be called to manipulate the files in the project. Here is some code that accesses key information from a file in the project:

```
WITH _vfp.ActiveProject
   ? .Files[1].Description  && Displays description of first file
   ? .Files[1].Modify()     && Modifies the first file in native designer
ENDWITH
```

Both the project and file object's properties can be set just like any other VFP object property through an assignment statement:

```
_vfp.ActiveProject.Files[1].Description = .T.
_vfp.ActiveProject.Files[1].Exclude = .T.
```

Through these two COM objects, we have almost complete control over all the files in our VFP projects. It is recommend that you investigate the sample program to gain more knowledge on these two COM interfaces into the VFP Project Manager.

Forms File (.SCX/SCT) and Visual Class Library File (.VCX/VCT)

The Forms table and the Visual Class Library table are identical in layout. They differ only in content. As you might already know, you can save a form as a class which is stored in the .VCX. There are only 23 fields in these metadata tables.

The .SCX contains the metadata for all forms and formsets. The .SCX files contain one record for each object on a Form, one header record for each form (there can be more than one form in a formset, all stored in one SCX), and a “reserved” comment record as the last record for a form. The information in this table is instantiated into a form at runtime directly from the metadata.

The .VCX contains class definitions for Visual FoxPro classes. The .VCX contains at least two records for each class; one class record and one “reserved” comment record for each class. If the class is a container, then there is one object record for each of the objects within the container as well. For instance, a formset will contain at least one form; therefore there is a formset record, a form record, and the “reserved” comment record. A container object might have labels, textboxes, editboxes, and potentially other containers.

The following piece of code will select all the textbox objects in a form or visual class library which is opened with an alias of “metadata”:

```
SELECT * ;
   FROM metadata ;
   WHERE UPPER(BaseClass) = "TEXTBOX" ;
     AND !EMPTY(BaseClass) ;
   INTO CURSOR query
```

The following code will create a cursor called query that lists all the baseclass objects contained in the form or visual class library along with a count of each object type.

```
SELECT PADR(BaseClass,75) AS Class, Count(*) AS Count ;
   FROM metadata ;
   WHERE !EMPTY(BaseClass) ;
   GROUP BY 1;
   INTO CURSOR query
```

The opportunities to leverage this metadata are only limited by your understanding of the metadata and how well you can write SQL code.

Back a few years ago, when developing in FoxPro 2.x I typically wrote cross platform applications in DOS and Windows. As my customers started upgrading their equipment to Windows 3.11 capable machines we moved away from development on the DOS platform. The DOS platform records gradually became obsolete over time. During one sweeping revision we eliminated the DOS code that was artificially bloating the generated .SPRs and .EXEs. To do this you open up the .SCX and do the following code:

```
DELETE ALL FOR Platform = DOS
```

You might be noting that we cannot do cross platform development with VFP these days. However, a number of developers are currently working on projects that are converting 2.6 applications to VFP. Conversions of this type which I have taken part in often leverage the reports from the old application (after all the report designer has not been upgraded significantly since the 2.0 days). These reports, whether cross platform between DOS and Windows, or just

DOS can be used in VFP applications. The transporter will translate the objects to the Windows platform in VFP. The above command to delete the DOS platform has come in handy in reducing the size of the application by removing these records.

Reports (.FRX/FRT) and Labels (.LBX/LBT)

The report and labels are identical just like the forms and visual class library files. The layout for these two tables weighs in heavily at 75 fields in VFP 5/6/7. The number of fields is a bit overwhelming at first, but you quickly see how it is used. Most of the fields are unused the majority of the time.

The key to leveraging report metadata files is knowing the different object types stored in the ObjType field:

1	Report	5	Text	9	Bandinfo	21	Printer Driver Setup
2	Workarea	6	Line	10	Group	25	Data Environment
3	Index	7	Box	17	Picture	26	Cursor Object
4	Relation	8	Repfield	18	Variable		

For instance, if you were looking for all the RepField objects in a report you could run the following code:

```
SELECT * ;
   FROM reporttable ;
   WHERE ObjType = 8 ;
   INTO CURSOR query
```

You can browse the table, or send the output to a custom report. Like forms and visual class libraries, reports often need to be reviewed by the development staff. One nice tool would create output of the different objects in a report for these reviews.

Another idea might be to get a summary count of all the objects in a report:

```
SELECT ObjType AS Class, Count(*) AS Count ;
      FROM metadata ;
      GROUP BY 1;
      INTO CURSOR query
```

Applications are distributed to customers who have a variety of preferences. One of the items that customers most like customized to their tastes is report layouts and fonts. While developers can make a good living changing and tweaking reports, there frequently is little time to handle this aspect of our jobs and these changes often become a lower priority.

The problem with the Report Designer when it comes to changing fonts is that you must pick not only the font, but the size, and if it is bold or italic for the report object. This means you must select objects that have the exact same font attribute, otherwise they all become the same. This is a very tedious process for reports that have a lot of different font sizes and styles. What if there was a way to speed up the process?

The report characteristics are stored in table format. The FontName is stored in the FontFace column of this table. The solution is to scan the table and replace the value of the column with a new FontName. Here is the core code of the ChgFont.prg program:

```
USE (tcReportMetadata) EXCLUSIVE IN 0 ALIAS curFontChanger

SELECT curFontChanger

SCAN
```

```
   IF !EMPTY(FontFace)
      REPLACE FontFace WITH tcFontName IN curFontChanger
   ENDIF
ENDSCAN

USE IN curFontChanger
```

There are other situations like multiple fonts being used in the same report and you may only want to change specific ones, for example:

```
REPLACE ALL FontFace WITH "Arial" FOR FontSize = 10
```

There is a danger to be aware of with this process. Different fonts have different font metrics. What this means is, different fonts have different heights, widths, shapes, sizes, and styles. This can have unexpected side affects if not used appropriately. If you are using a plain fixed size font like Courier New and decide to change to something like Goudy Stout, your label objects may get truncated since Visual FoxPro stores the pixel width of the text objects. The sizes in the development designers are not reflective of what you get when you run the report with a `REPORT FORM` command. If a font specified in a report is not loaded on the user's PC, Windows will substitute one that it feels closely represents the same attributes as the font you specified. Windows is not always good at this and the report may print differently than designed. On the other hand, if you are moving to fonts with similar font metrics you will save a lot of time and effort pleasing the customer.

Menus (.MNX/MNT)

The VFP Menu metadata file is the only file that is used to generate code to the program (into a .MPR) format. This is still done using GenMenu.prg which is included in VFP. There were some changes related to the new shortcut menus available in VFP 5.0. The file layouts changed again in VFP 7 because of the new icon capability that was added. The SysRes and ResName columns were added. As I noted previously, this table is the only metadata file in the mix that does not have the famous USER field which is available for the Visual FoxPro developer to use to their hearts content.

One interesting concept with menus is that the metadata is “upside down”. The records in the table are from the bottom up, and from right to left. One might expect that the records in the table are from the first pad to the last pad, and the bars from one to the number on the pad. This is definitely not the case so keep this in mind when working with the menu metadata.

Database Container (.DBC/DCT/DCX)

The Database Container is the only metadata table that has an index file associated with it. It provides VFP developers with a data dictionary. It is also the simplest of the metadata files as far as layout. All the information contained within the database can be extracted using the VFP DBGetProp() function. See VFP Help for all the details.

I have leveraged the Database Container for documentation in a program called DBCInfo.prg. This program is included in the Project Lister and as part of the sample code included with the files from this session. Here is a portion of DBCInfo.prg as an example of what can be done with the DBC. This code returns specific information about the tables contained in the database:

```
tlDetails  = .T.
lcDBCNotes = ""
lcDatabase = GETFILE("DBC")
```

```
OPEN DATABASE (lcDatabase) SHARED

lnTables      = ADBOBJECTS(axTables,"TABLE")

=ASORT(axTables)

IF lnTables > 0
   lcDBCNotes = lcDBCNotes + "Tables:" + CHR(13)

   FOR lnCounter = 1 TO lnTables
      lcExtendedDetails = ""

      IF tlDetails
         lcTableComment     = DBGETPROP(axTables[lnCounter], "TABLE", "Comment")
         lcTablePath        = DBGETPROP(axTables[lnCounter], "TABLE", "Path")
         lcTableUpdate      = DBGETPROP(axTables[lnCounter], "TABLE", "UpdateTrigger")
         lcTableDelete      = DBGETPROP(axTables[lnCounter], "TABLE", "DeleteTrigger")
         lcTableInsert      = DBGETPROP(axTables[lnCounter], "TABLE", "InsertTrigger")

         lcExtendedDetails  = lcExtendedDetails + ;
                              IIF(EMPTY(lcTableComment),"","Comment:"+lcTableComment+", ")
         lcExtendedDetails  = lcExtendedDetails + ;
                              IIF(EMPTY(lcTablePath),"","Path:"+lcTablePath+", ")
         lcExtendedDetails  = lcExtendedDetails + IIF(EMPTY(lcTableUpdate),"",CHR(13)+;
                                                      "Update Trigger:"+lcTableUpdate+", ")
         lcExtendedDetails  = lcExtendedDetails + IIF(EMPTY(lcTableDelete),"",CHR(13)+;
                                                      "Delete Trigger:"+lcTableDelete+", ")
         lcExtendedDetails  = lcExtendedDetails + IIF(EMPTY(lcTableInsert),"",CHR(13)+;
                                                      "Insert Trigger:"+lcTableInsert+", ")

         * Remove trailing comma
         lcExtendedDetails  = SUBSTR(lcExtendedDetails, 1, LEN(lcExtendedDetails)-2)
      ELSE
         lcTablePath        = DBGETPROP(axTables[lnCounter], "TABLE", "Path")
      ENDIF


      lcDBCNotes = lcDBCNotes + ALLTRIM(STR(lnCounter)) + ". " + ;
                   axTables[lnCounter] + ;
                   IIF(EMPTY(lcExtendedDetails), "", " - ") + lcExtendedDetails + CHR(13)
   NEXT lnCounter

   lcDBCNotes = lcDBCNotes + CHR(13)
ELSE
   lcNoInformation = lcNoInformation + "No Tables" + CHR(13)
ENDIF

RETURN lcDBCNotes
*: EOF :*
```

One of the better programs that leverage metadata from the DBC is GenDBC.prg. This utility is included with Visual FoxPro. GenDBC will generate the code to build the database, contained tables, views, including all the rules, input masks, comments, etc. The files will be empty, but the database structure is recreated from scratch. This is ideal for an application initial install or setting up a test system with empty files. GenDBC can be found in the \VFP\TOOLS\GENDBC\ directory.

Visual FoxPro Tool Metadata

There are four tables that VFP uses interactively as you use the development environment. While these might not be source code metadata, it is certainly metadata about the development environment. As a developer you can expose these tables in your developer tools.

FoxCode.dbf This table is where all the IntelliSense information is stored for use in the

	IDE when you have IntelliSense turned on.
FoxTask.dbf	This table contains all the items you have stored in the TaskList.
Browser.dbf	This table stores all the Class Browser and Component Gallery add-ons.
FoxRefs.dbf	This table tracks all the COM objects that are registered on the computer for use by the Object Browser.

Sources of Documentation

Documentation Projects

There are several projects included with Visual FoxPro that detail the metadata table file layouts. This would be metadata about metadata <g>. These projects are located in the \VFP\FILESPEC directory. Visual FoxPro 5/6/7 includes the layouts for FoxPro 2.6 and VFP. Unfortunately each version of VFP does not include the layouts for previous versions of VFP. The files names are:

- 26SPEC.PJX
- 30SPEC.PJX (With VFP 3 only)
- 50SPEC.PJX (With VFP 5 only)
- 60SPEC.PJX (With VFP 6 and VFP 7, metadata descriptions are up to date to VFP 7)
- 70SPEC.PJX (Does **not** exist with VFP 7 (bug!), see the 60SPEC.PJX)

There are several reports in each of the projects. Here are the files that are documented in the 60SPEC.PJX as they ship with VFP 7.

	Type	File Name	Updated	Description
	Project File	60SPEC.PJX	06/17/1996 02:19:54 PM	Directory is: C:\PROGRAM FILES\MICROSOFT VISUAL FOXPRO 7\TOOLS\FILESPEC\ Debug code created, Shows FoxPro logo at start up, Comment style is box, Generated code is not saved, No Code Page
	Report File	60PJX1.FRX	06/14/1996 06:43:48 PM	Directory is: C:\PROGRAM FILES\MICROSOFT VISUAL FOXPRO 7\TOOLS\FILESPEC\ No Code Page
	Report File	60PJX2.FRX	06/17/1996 02:19:28 PM	Directory is: C:\PROGRAM FILES\MICROSOFT VISUAL FOXPRO 7\TOOLS\FILESPEC\ Code Page:1252
	Report File	60DBC.FRX	06/14/1996 07:04:54 PM	Directory is: C:\PROGRAM FILES\MICROSOFT VISUAL FOXPRO 7\TOOLS\FILESPEC\ Code Page:1252
	Report File	60DBCPRO.FRX	06/14/1996 06:38:30 PM	Directory is: C:\PROGRAM FILES\MICROSOFT VISUAL FOXPRO 7\TOOLS\FILESPEC\ Code Page:1252
	Report File	60FRX1.FRX	06/14/1996 06:40:44 PM	Directory is: C:\PROGRAM FILES\MICROSOFT VISUAL FOXPRO 7\TOOLS\FILESPEC\ Code Page:1252
	Report File	60FRX2.FRX	06/14/1996 06:40:14 PM	Directory is: C:\PROGRAM FILES\MICROSOFT VISUAL FOXPRO 7\TOOLS\FILESPEC\ Code Page:1252
	Report File	60MNX.FRX	06/14/1996 05:48:14 PM	Directory is: C:\PROGRAM FILES\MICROSOFT VISUAL FOXPRO 7\TOOLS\FILESPEC\ Code Page:1252
	Report File	60SCX1.FRX	06/14/1996 06:52:22 PM	Directory is: C:\PROGRAM FILES\MICROSOFT VISUAL FOXPRO 7\TOOLS\FILESPEC\ Code Page:1252
	Report File	60SCX2.FRX	06/14/1996 07:02:52 PM	Directory is: C:\PROGRAM FILES\MICROSOFT VISUAL FOXPRO 7\TOOLS\FILESPEC\ Code Page:1252

This is by far the best documentation about metadata provided with Visual FoxPro. These reports look like the detail listings that were included in the 2.x Developer's Guide Appendix A. To access the documentation you can either open the free tables and read the memo fields, or just preview the reports and print out the documentation you need.

FoxPro KnowledgeBase (Microsoft Developer Network (MSDN))

The FoxPro KnowledgeBase is a collection of articles written by developers and Microsoft employees that feature topics about FoxPro 2.x and Visual FoxPro. These articles highlight bugs, provide workarounds, discuss new features in depth, and are a general good source for information on VFP.

The KnowledgeBase can be searched by keyword. I used keywords like “metadata”, “structures”, “vfoxwin”, and “6.00”. I prefer to use my MSDN CD since I can search all day if I want without connect time issues. The online version (on the Internet) are updated and have the opportunity to be most current.

There are excellent articles in the FoxPro KnowledgeBase concerning metadata. “How to Find Out Which FoxPro Version Created an .SCX File’ (Q129572) and “File Structure and Contents of .VCX and .SCX Files” (Q145742) are a couple of fine examples. The Visual Class Library is documented as well as I have ever seen it in the FoxPro KnowledgeBase in the “Details of the .VCX File Architecture” (Q130085) article. Each of the fields in the .VCX/SCX are discussed in detail. Information in the article “Details of the .VCX File Architecture” was obtained from “Visual FoxPro's VCX and SCX Architectures” written by Ken Levy in the March 1995 issue of *FoxTalk*.

Online Help and Online Documentation

The Online Help and Online Documentation are really poor sources of information on metadata. I searched for “metadata”, “meta”, “projects”, “PJX”, and “SCX”. The only topics worth anything on metadata were the “Table Structures of Table Files” topic and this points you to the above described VFP projects.

FoxPro 2.x Developer’s Guide (Appendix A)

This is an excellent source for FoxPro 2.0, 2.5 and 2.6 metadata file layouts. Most of the discussion in this presentation was for Visual FoxPro, but sometimes you will continue to work on a 2.x project or inherit another developer’s project and you might want to leverage some of the metadata to get a head start.

Cool Tools that Use Metadata

Please note that all the tools discussed here are tools primarily developed for use by the developer (a very common case). The code may not be optimized, was not written to customer specs, and in some cases not designed and written particularly well. Andy Neil wrote an article titled “Between a Rock and a Hard Place: Application Developer Solutions” in the December 1996 of *FoxPro Advisor*. In this article Andy discusses many tradeoffs made when developing solutions that aid developers to build customer based solutions. The fact of the matter is, not a lot of care is given to the code from a standards and optimization level as long as it gets the job done. Great care was given to the code from a quality standpoint; these tools are bug free as far as I know. I use most of them weekly, if not daily. If you encounter something that appears fishy while using these tools please drop me a note, my Internet Id is on the title page of this document. I will look into it.

Metadata Clean Up

Examples: PackMeta.prg (in downloads)

It is important to remember that the data in the metadata tables are very volatile. We typically open a visual class library or form to make changes. These changes are written to the table and memo file. Just like .DBF tables and associated .FPT files, we get "memo bloat". Memo fields are always added to the end of the memo file when added new or modified. This makes it easy for FoxPro to grab the memo field all at once, not from different parts of the file. To accomplish this, the first copy of the memo is logically deleted and the pointer in the .DBF field is updated to the newly appended location. This trade off for speed and efficiency can create very large memo files with plenty of obsolete data. This is especially true if you are constantly tweaking the code. All the properties and methods reside in these memo fields in the metadata table.

Figure 2. *The Class Browser has a built in PACK method to remove memo bloat from metadata.*

So how do you clean up the tables? There are two options. The first is VFP Class Browser. Open up the .VCX or .SCX using the Class Browser and then press the Cleanup commandbutton. This method is fine if you have one or two files.

The second method is to perform a Rebuild All when you are building the project.

The third method takes advantage of the VFP metadata table by opening a project and scanning the file for forms, visual Class Libraries, reports, menus, etc. Here is the guts of a more complete program called *PackMeta.prg* (included with session files):

```
SCAN
   SCATTER MEMVAR MEMO

   * Count Subdirectories so actual directory name can be printed

   nSubDir = 0
   nIndex  = 1

   DO WHILE .T.
      DO CASE
         CASE SUBSTR(name, nIndex, 4) = "..\\"
            nSubDir = nSubDir + 1
            nIndex  = nIndex + 4
         CASE SUBSTR(name, nIndex, 3) = "..\"
            nSubDir = nSubDir + 1
```

```
            nIndex  = nIndex + 3
         OTHERWISE
            EXIT
      ENDCASE
   ENDDO

   * Process "calculated" fields, remember mNotes is based on parameter
   DO CASE
      CASE nSubDir = 0 AND SUBSTR(Name,1,1) = "\"     &&  Absolute directories
         cParentDir = SUBSTR(cHomeDir, 1, 2)
      CASE nSubDir = 0 AND SUBSTR(Name,2,1) = ":"     &&  Complete path
         cParentDir = ""
      CASE nSubDir = 0 AND SUBSTR(Name,1,1) != "\"    &&  Relative directories
         cParentDir = cHomeDir

      OTHERWISE                                    && One or more parental directories
         cParentDir = UPPER(SUBSTR(cHomeDir, 1, RAT("\",cHomeDir, nSubDir+1)))
   ENDCASE

   m.mDirName = cParentDir + UPPER(SUBSTR(Name, nIndex , RAT("\",name)-nIndex+1))

   lcMetaData = m.mDirName + ShortNamePR(Name)

   USE (lcMetaData) ;
       IN SELECT(1) ;
       ALIAS MetaData EXCLUSIVE

   SELECT MetaData
   WAIT WINDOW PADC("Packing: " + lcMetaData, 80) NOWAIT
   PACK
   USE

   SELECT projtemp

   lnRecordsProcessed = lnRecordsProcessed + 1
ENDSCAN
```

Another method (and less complicated) could include packing metadata by directory. This would be done by processing an array created by ADIR().

```
=ADIR(axMetaData, "*.VCX")
=ADIR(axMetaData, "*.SCX")
=ADIR(axMetaData, "*.FRX")
=ADIR(axMetaData, "*.LBX")
=ADIR(axMetaData, "*.MNX")
```

Then you would loop through the arrays, open the metadata table and PACK. This code is much simpler, but I like to work from a project perspective.

Code Review Listings

Examples: PrintCX.prg, PrintFrx.prg, PrintMnx.prg (in downloads)

One of the easiest ways to insure that code is developed optimally is to have other developers review the finished product. The code is reviewed before we go into final User Acceptance Testing. I personally like Whil Hentzen's reference to these sessions in his book "The 1999 Developer's Guide" as "Defend Your Life".

In the past (meaning FoxPro 2.x) we had plenty of code to review. The obvious code is stored in .PRGs. We also had generated code in .SPRs and .MPRs. All the stored metadata was reviewable in the generated code form. With the advent of Visual FoxPro the code for forms and the visual class libraries are no longer generated and therefore more difficult to review. I created the Code Review Listing (PrintCX.prg) to list off the code in a way more conducive to a team code review.

The VFP Class Browser does a good job at “exporting” the code for forms and visual class libraries, but frankly misses a few items and formats the code in a way I find difficult to review. The biggest missing item when exporting form code is the dataenvironment. For those developers that use the native dataenviroment, this can be important to review. The Class Browser also lower cases all object names that I worked so hard typing in with hungarian notation. The final reason I wrote this handy program is the object order. I placed the objects in container order, then object name order. This is more logical for me to review the code.

A future enhancement idea includes walking up the class hierarchy for those objects that inherit code.

The Report Code Lister provides a printout of the details inside of the report form metadata file. You are prompted for a FRX (report metadata table), and then the program evaluates the different objects. The listing presents the code (methods in the dataenvironment or fields on the report) and attributes for each object as well as some other details stored inside the metadata. The objects are listed in the order they appear on the report from top to bottom and left to right. This tool was designed to allow teams to review code developed by other developers on their team. The output is presented in a report preview so the developer can check it out before printing the report to the printer. There is no tool in VFP that lists code stored in the report form.

The Menu Code Lister provides a printout of the details inside of the Menu metadata file. You are prompted for a MNX (menu metadata table); the program evaluates the information in the metadata table, and then lists off each of the menu options. Information included is Pad Prompts, Bar Prompts (along with relative position on the drop down), Status Bar Messages, hot keys, procedure code (setup, cleanup, and procedures for each bar), relative positioning of Menu Pads, and bars names.

One might note that Visual FoxPro still generates the program code for the menu via GenMenu.prg and that the code is very readable for a walkthrough. This is true. The RAS PrintMNX does not attempt to duplicate this technique. It provides a "graphical" representation of the menu for the developers to look at and evaluate with the core code next to it.

Note that each of these programs have #DEFINEs in them at the beginning of the program to indicate which reports and metadata engine to use when creating the output. This was done with hard coded, full paths, so the programs could be run with any default directory.

Internationalizing an Application

The following tool is not included in the files available from this session since it is pretty much very specific to the job at hand at the time. The idea is presented here to inspire other developers to leverage the most simple ideas to save tons of time.

One application I designed and developed is one nearly every database developer has tackled at one point in their career, an Address Book. This particular Address Book tracked all the branch office sites of a global company, the area and region hierarchy, and high level management names and phone numbers. Numerous validation tables, reports, and mailing labels are also included. The system was designed to run in 300 sites in the United States. One day the customer decided they wanted to install the application in sites in Quebec, Canada and asked what it would take to run the application in French. “No problem” I said and quickly gave them an estimate. Fortunately we had purchased Steven Black’s INTL ToolKit.

Each screen needed to be modified for international features, all 30 or so. I also wanted to change all the screen fonts from MS San Serif to Arial. The first few screens were painfully modified. Then, I wrote a simple program that opened up the SCX, changed all the fonts to Arial for all objects, and inserted the needed hooks for INTL. The entire application was converted, tested and implemented in a 2 week time frame. The tool took less than an hour to write and test and has saved months worth of work.

Extending the IDE

IntelliSense

It is beyond the scope of this session to discuss IntelliSense at length, but it should be noted that you can really extend the capabilities of the editor with IntelliSense.

Class Browser add-ins

Example: CBChangeDateFormat.prg (in downloads)

Class Browser add-ins allow developers to extend the functionality of the native VFP Class Browser. You are basically snapping in extra code that the Class Browser knows about when it starts up. You can even have it run code when it starts.

One example that I have written is to have it show the same date format as I have set in my Windows Regional Settings. My preference is to show the full century. The Class Browser defaults to `SET CENTURY OFF`. With this in mind I wrote the following code:

```
LPARAMETERS toBrowser

LOCAL lcName                           && Name of the Add-in
LOCAL lcComment                        && Comment for the Add-in

* Self registration if not called form the Class Browser
IF TYPE("toBrowser")= "L"
   lcName    = "Rick Schummer's Date Setting Changer"
   lcComment = "Developed by RAS for online forum discussion and example"

   IF TYPE("_oBrowser")= "O"
      * If Class Browser is running, use Addin() method
      _oBrowser.Addin(lcName, STRTRAN(SYS(16),".FXP",".PRG"), "ACTIVATE", , , lcComment)
   ELSE
      * Use the low level access of the Browser registration table
      IF FILE(HOME() + "BROWSER.DBF")
         lcOldSelect = SELECT()

         USE (HOME() + "BROWSER") IN 0 AGAIN SHARED ALIAS curRASDateChanger
         SELECT curRASDateChanger
         LOCATE FOR Type = "ADDIN" AND Name = lcName

         IF EOF()
           APPEND BLANK
         ENDIF

         * Always replace with the latest information
         REPLACE Platform WITH "WINDOWS", ;
                 Type     WITH "ADDIN", ;
                 Id       WITH "METHOD", ;
                 Name     WITH lcName, ;
                 Method   WITH "ACTIVATE", ;
                 Program  WITH LOWER( STRTRAN( SYS(16), ".FXP", ".PRG")), ;
                 Comment  WITH lcComment
         USE
```

```
         SELECT (lcOldSelect)
      ELSE
         MESSAGEBOX("Could not find the table " + HOME() + "BROWSER.DBF" + ", please make sure it
exists.", ;
                    0 + 48, ;
                    _screen.Caption)
      ENDIF
   ENDIF

   RETURN
ELSE
   * Check to see if we really got called from the Class Browser
   IF !PEMSTATUS(toBrowser, "lFileMode", 5)
     RETURN .F.
   ENDIF

   * Now the simple stuff to change the Date environment
   * setting which is specific to the private datasession.
   * This setting is driven from the developer's Windows'
   * Regional Settings.
   SET SYSFORMATS ON

   * Then refresh the Class Browser to reflect the change
   toBrowser.Refresh()
ENDIF

RETURN

*: EOF :*
```

The majority of the code performs a self registration to the Browser.dbf. The base of the code that runs each time is the “else” of the main “if”. It checks to see if the code was run from an instance of the Class Browser (CB), then does a SET SYSFORMATS ON and refreshes the CB form. It is not in the scope of this session to explain all there is to know about the Class Browser add-in, but rather to show how easy it is to write your own.

Builders

Example: GenProjectHook.prg (in downloads)

Builders are tools that aid in setting properties of objects. Unlike wizards, builders are re-entrant, so you can open a builder on a particular object again and again. VFP includes a number of generic builders for many of the common form objects including grids, list boxes, combo boxes, check boxes, command groups, and option groups.

Builders do not have to work with the Form or Class Designer as you work interactively with the objects. You can write code that creates classes or forms and literally builds these objects from scratch. It is the same technology. You can write code that will open an existing class and alter a number of objects for a specific property. You can write code that will convert various form objects to your new framework baseclasses or enforce some standard within your company.

One example of using some of the Builder Technology is to generate a class and set the property. Here is an example that has a program create a projecthook class and then set one of the properties:

```
LPARAMETER tcProjectHook, tcProjectFieldMappingConfig

#DEFINE ccSYNTAX           "do GenProjectHook with tcProjectHook, tcProjectFieldMappingConfig"
#DEFINE ccPROJECTHOOKSLIB
"D:\Data\WinWord\Presentations\GLGDW2000\FishingWithProjectHook\PHookSamples\cPhkDevelopment"
```

```
#DEFINE ccPROJECTHOOKSBASE "phkCompanyStandard"

LOCAL laProjectHookRef[1], ;           && Array that contains a reference to projecthook created
      lnClasses, ;                     && Number of classes in the projecthook classlib
      llFound, ;                       && Flag to determine if class already exists
      lnCounter, ;                     && For loop counter
      lcOldSafety, ;                   && Save the Safety setting
      lcOldAlternate, ;                && Save Alternate setting
      lcOldAlternateFile, ;            && Save Alternate,1 setting
      lcOldCentury                     && Save Century setting

tcProjectHook               = ALLTRIM(tcProjectHook)
tcProjectFieldMappingConfig = ALLTRIM(tcProjectFieldMappingConfig)

lcOldSafety        = SET("safety")
lcOldAlternate     = SET("alternate")
lcOldAlternateFile = SET("alternate",1)

SET SAFETY OFF

SET ALTERNATE TO FULLPATH(CURDIR()) + "GenProjectHook.txt" ADDITIVE
SET ALTERNATE ON

lnClasses = AVCXCLASSES(laClasses, ccPROJECTHOOKSLIB)
llFound   = .F.

FOR lnCounter = 1 TO lnClasses
   IF LOWER(laClasses[lnCounter, 1]) == LOWER(tcProjectHook)
      MESSAGEBOX(tcProjectHook + " projecthook was not created because it already exists", ;
         0 + 64, ;
         _SCREEN.CAPTION)

      ? "ProjectHook " + tcProjectHook + " skipped because it existed at " + TTOC(DATETIME())
      RETURN .F.
   ENDIF
ENDFOR

? "ProjectHook created " + tcProjectHook + " at " + TTOC(DATETIME())

* Close the log
SET ALTERNATE OFF
SET ALTERNATE TO

* Reset to the original settings
SET ALTERNATE TO &lcOldAlternateFile
SET ALTERNATE &lcOldAlternate

SET SAFETY ON

CREATE CLASS (tcProjectHook) OF ccPROJECTHOOKSLIB ;
   AS ccPROJECTHOOKSBASE FROM ccPROJECTHOOKSLIB NOWAIT

* Set the class property for the projecthook cFieldMappingCategory property
ASELOBJ(laProjectHookRef, 1)

IF TYPE("laProjectHookRef[1]") = "O"
   laProjectHookRef[1].cFieldMappingCategory = tcProjectFieldMappingConfig
ENDIF

* Make sure the reference to the projecthook is released
RELEASE laProjectHookRef

* Handle the VFP Windows that open with no Resource File
IF WEXIST("PROPERTIES")
   RELEASE WINDOW "Properties"
ENDIF

IF WEXIST("FORM CONTROLS")
   RELEASE WINDOW "FORM CONTROLS"
ENDIF

* Close the newly created class opened in Class Designer
```

```
* The keystrokes are "buffered" until all classes are created
KEYBOARD '{CTRL+W}'

* Added to close the class designers down
DOEVENTS()

* Reset the environment
SET SAFETY &lcOldSafety

RETURN .T.

*: EOF :*
```

Adding developer menus

Developer menus enable developers to add functionality to the IDE from the main menu. My developer menu has a list of often accessed tools (built, grabbed, and bought) that I used every day. I have the menu work as a list of shortcuts to not only tools but simple commands that I repeatedly type into the Command Window.

The developer menus are as easy to build as any custom application menu and I use the same underpowered Menu Designer as I used for the apps. I make sure of one key setting. The developer menu is a “before menu” so when I run it the menu always appears before the Windows menu pad.

Figure 3. *This is Rick Schummer’s developer menu.*

Macros (also works in production apps)

If I was asked to point to one feature in VFP that is under utilized or underrated, I would have to point to the Macro capability. The development environment is so open and so programmable that I think people forget about the Macro recorder and assignment to keys. Macros will be ignored even more with the advent of IntelliSense.

Before we had the indent capability available from the editor shortcut menu, I use to program macros to indent three spaces, hit the home key and arrow down to the next line. I could then hit the one key and indent the line instead of five keys. This is a simple idea that saves me time over and over.

Also, developers often forget that the Macro capability is available in the runtime environment; therefore you can expose this to the users of your killer applications. They can program keystrokes that might traverse a menu option, fill in a couple of fields and have the form set up to do the rest of the data entry.

Why Grab?

There are plenty of other VFP developers who struggle the same way you do and may have written tools first. Developers have different pain thresholds, but we all work with the same great development tool so we all hit the same weaknesses and have the same set of strengths take advantage of in our development. A number of other developers have identified weaknesses with VFP and have created home grown products to ease a specific pain, or more importantly, enhanced how they approach their development.

The biggest advantage of grabbing a developer tool is the cost. They are free for download from a website or one of the forum sites that have download sections. Most of the simple developer tools that I have built have more than a couple of hours into them. They easily save me the time when used again and again, but if I can grab the same tool for free and work on billable work I am way ahead financially.

The second reason to grab freebies is that the time already invested working out the bugs. Freeware products have an additional advantage in the fact that they have a base of developers already using the product. These developers have different needs for the products to support and offer enhancement requests and the tool developer might be kind enough to see these enhancements to their advantage and include them in later versions. This is not always the case though. The important thing is to get the source code for the tool if it is available. If the source is not available, make sure the developer is responsive to the changes. If neither, then consider the option to build.

Tools to Grab

ViewEditor

http://geeksandgurus.com/
http://stephensawyer.com/

The ViewEditor is the simply the best freebie developer tool available. This tool was written by Steve Sawyer to allow VFP developers to scrap the underpowered VFP View Designer. It provides developers who are comfortable writing SQL selects, but dislike writing all the accompanied `DBSETPROP()` statements with a tool to quickly assemble complex views, both local and remote.

The VFP View Designer has been great as a simple learning tool for simple queries. It falls apart the moment you have two child tables from one parent. This basic problem has forced developers to another view creation strategy. The common approach is to keep the views in a program via the `CREATE SQL VIEW` command. This can be tedious because of all the `DBSETPROP()` commands that are associated with the view. Other developers have created solutions (Erik Moore's EVIEW comes to mind) to modify properties of existing views, but never delivered the complete package.

Figure 4. *The View Editor completely replaces any need for the VFP View Editor.*

Steve's ViewEditor allows developers to create and modify local and remote views. All of the property settings that you have available in the VFP View Designer are also exposed. There are also some features not found in the View Designer like the ability to fix the field UpdateName property and the ability to migrate table field properties to the view field properties. The ViewEditor will also generate the code for the view in the same fashion as GenDBC for the view currently edited. Other tools like Stonefield Database Toolkit will fire up ViewEditor instead of the native View Designer if you configure it to do so.

G2 SuperCls

http://geeksandgurus.com/
http://rickschummer.com/vfptools.htm

Most Visual FoxPro developers are familiar with Ken Levy's SuperClass utility. This utility works as a visible toolbar whenever the method editor is opened. The first toolbar button (see Figure 5) allows you to edit the code in the superclass (parentclass). You cannot open the superclass to edit code when the subclass is already open in the Class Designer. This tool provided developers with a way to peak at the code to see what they might be completely overriding or to determine how to extend behavior with a call to the superclass via a `DODEFAULT()` or a call up the class hierarchy via a direct call via the Scope Resolution Operator (::). Right-clicking on the Edit Superclass button will show all the classes in the hierarchy so you can select which level of code to review.

Figure 5. *Original Superclass Utility by Ken Levy.*

Figure 6. *Upgraded Superclass Utility from Geeks and Gurus.*

Drew Speedie, architect of Visual MaxFrame Professional extended this tool with an additional button. The functionality he added displayed all the code in each of the superclasses in one edit window. This code cannot be edited in any fashion, but it quickly allowed the developer to see all the code that would be executed provided that the appropriate calls were made in the hierarchy. However, this tool is shipped with VMP, and requires code in VMP to run. Since my partner Steve Sawyer and I are fond of Drew's additional feature, but we are using another framework at the moment, we decided to hack Ken's source code and plug in Drew's functionality in a generic manner, one that is independent of the VMP framework code.

Figure 7. *Output of the code in the class hierarchy is displayed in a program editor for colorization.*

HackForm

http://geeksandgurus.com
http://rickschummer.com

Have you ever renamed a Visual Class Library? I can already hear the groans and see the looks of terror on the faces of all the developers who know this pain.

What happens next? Try opening a Form that has objects you renamed in the class library. "Error instantiating class. Cannot find <classname> in <class library>". Sure you can locate the renamed classes and the form is displayed in the Form Designer. Close the form, open it up again and see that the locate dialog is presented again. Even though you pointed the object to the right class the Form Designer never noted the form changed to kick in the Save logic when closed down. Nice, huh? Sounds like a bug to me. Make another change to the form to work around this.

Even more aggravating than this is that every object that was renamed (or if the class library was renamed) will spawn the locate dialog. The same dialog is displayed over and over even if the new class or class library were located before. This can be quite frustrating after the first 5 or so changes.

Have you ever wanted to change the baseclass of an object, but the BaseClass property is protected in the VFP Property Sheet? The VFP Class Browser will allow you to change the BaseClass of the outer-most container or class, but it will not allow you to change the contained class.

Figure 8. *Hacking a VCX the old fashioned way.*

So what happens when something does go wrong and you just need to get inside the .VCX or .SCX? One can USE the file and browse it. Unfortunately all the stuff you want to see is in memo fields. Now you have to start looking for all the fields you want to look into, resize them, move them to another place so the information is viewable, etc. The next time you open it up with a BROWSE LAST you will be okay, but a new class will cause you this grief all over again!

Figure 9. *The RAS HackForm allows you to change almost any property, method, or VFP protected settings in the class or form.*

I built the Hack VCX/SCX form to get around some of the configuration frustrations that came along with the BROWSE technique. This form allows you to select a Form or Visual Class Library. It opens it and displays the information inside the file. I protected some of the fields I felt I had no business changing at all. The rest are open to modify to my hearts content (or until I can disable the file all together).

RAS ProjectBuilder

http://geeksandgurus.com
http://rickschummer.com

The RAS Project Builder provides an interface that consolidates some of the Project options from the VFP Project Information dialog, some of the VFP Project Builder and Version dialogs, as well as some handy hooks to the RAS ProjectHook (if instantiated), and resolves some issues with SET STRICTDATE for production builds.

This tool was designed to handle some of the my ideals when building development versions and building the "gold" code for production releases.

Feature	Development	Production
Recompile All Files	Default OFF	Default ON
Display Errors	Default OFF	Default ON
Debug Code	Default ON	Default OFF
SET STRICTDATE TO	2	1
Clean Printer Code from reports (FRXs) (note this is only available with RAS ProjectHooks)	Default OFF	Default ON

All the settings that are retained in the project file (Auto Increment, Encrypted Executable, and Version page) are read from the settings in the project. The Process Project Audit Trail, Clean Printer Information from Reports, Project Activate (only for VFP 6), and Field Mapping features do require the RAS ProjectHook activated for the project (included in the same file, just not required).

Figure 10. *The RAS Project Builder saves time toggling between the various VFP Build Dialogs.*

GetConstants

http://www.west-wind.com/webtools.asp

Right from the West Winds website:

"Have you ever wanted to have an easy way to access those COM constants that are so easily accessible in VBA via Intellisense? Like the ADO or MSXML or MSMQ constants that are used in all the MSDN samples, but never shown with their actual values? Well this simply utility lets export all of those constants into a single header (.h) file which you can then simply include into your programs to access those same constants in your code without Intellisense."

This tool has become less useful with the advent of the VFP 7 Object Browser, but if you have been living under a rock or have a pointy haired boss or a under educated technical lead that refuses to let you upgrade to VFP 7, go grab this tool. It is even handy for developers using VFP 7. Before these tools were available you had to grok the constants using an object browser like the ones include in the VBA editors in Microsoft Office or had to scramble using some tool to look into the type library file for the COM object. This automated process is so much better.

Project Search

http://stevedingle.com/

Steve Dingle's Project Search tool is very handy when you need to search for a specific text string and want to check all the files in a specific project. Naturally, the Windows Explorer will search for text, but it is directory centric. The result set generated allows developers to open the source code (even searches the VFP metadata files like forms, visual class libraries, and reports) directly into a method that the search string resides.

Figure 11. *Project Search looks for string values in files contained in a project.*

Project Search is very configurable and extendible. You can write your own output mechanism so if you can think of something different from what Steve delivers you as a developer can put it together. There is even a help file included.

Go Fish

http://www.universalthread.com

This tool will search directories or projects and look for a text string. Plain and simple. You can configure which file types are searched. The difference of this tool vs. other tools that provide this functionality is that this tool also allows you to replace text. Besides the automated replace functionality, you can also edit the text directly.

Figure 12. *Interface of Go Fish v2.0.*

Project Lister

http://geeksandgurus.com
http://rickschummer.com

The Project Lister is the oldest of RAS crafted developer tools. The utility was developed like so many others, out of frustration. I like to make checklists when processing through projects. A checklist for forms that need to be processed in an update, another for code that needs to processed for a walk through, and so on. All objects in a project are already stored in the FoxPro Project file (.PJX).

At first the program started out listing project objects by name, and then I needed other sort orders. I built a report to better present the output, then I wanted only specific objects so I added a table to drive the selection of object types. Then I wanted these objects in a specified order, etc., etc., etc. Each option became a new parameter to the program.

Then one day (by accident since this was a secret tool <g>) a teammate saw some of the output on the printer and wanted to be able to use the tool. It was difficult to use with all the parameters unless you had them memorized like I did. I added an interface, some new output options, and a help system when I decided to make it available on CompuServe and presented the idea in an article titled “FoxPro’s Project File” in the November 1994 issue of *FoxTalk*. This fine example of project scope creep is very typical when developing tools.

Figure 13. *Interface of the Project Lister.*

Figure 14. *One of the reports provided by the Project Lister shown in Acrobat.*

This utility can be used for more than check lists. I use the “Copy to Clipboard” capability to export information to a format that is easily translated to a MS Word table in documentation. I like the Database Container documentation that is generated from the Project Lister as well. This DBC documentation can be run separately as well using the DBCInfo.prg which is included in the Project Lister project.

Features:

- Select project to document from an open project or one residing on disk
- Configurable reports allow you to select what information appears
- Configurable list of project objects allow you to select what object types appear on reports
- Can browse the objects that will appear in output
- Select from several reports (both large and compressed print) and different orders

- Output to preview report, printed report, text file, Windows' clipboard, or free table format
- Requires VFP 6.0 or higher because it incorporates some of the new Project Object properties, events, and methods.

Task List Replacement

http://hentzenwerke.com (once *1002 Things You Wanted To Know About Extending VFP* is released)
http://geeksandgurus.com
http://rickschummer.com

Microsoft has put in place enough hooks to completely build our own Task List Editor. So we decided to put some of this new found knowledge to work in a tool called the G2 Task List Editor and address some of the limitations of the Visual FoxPro Task List.

You can add (user defined and other tasks), update, and delete tasks. We decided not to add shortcuts task types since it is much easier to add tasks from the various code editors. The data is "record buffered" since all the data is accessed via the task object data. It can be reverted until you move to another task. You can accomplish the same thing in the Visual FoxPro Task List if you edit the data in the Task Property dialog, but the "grid view" of the tasks has no way to revert any changes.

There are a couple of features that are included in the G2 version that are not included in the one that ships with Visual FoxPro. The first issue addressed is that the native Task List does not display the existing information in the user-defined columns in the Task Properties. The example ships with _Developer and _Comments extended properties exposed. This version of the tool also provides a mechanism to have default values for the user-defined columns in the various add methods. The native tool does not have a mechanism to add "other" task types, this can be accomplished by clicking on the "Add Other" task button.

There is a known issue with the initial release. If you expose the Task List interface and then shut it down, you lose the reference to the task list engine which causes problems with the G2 Task List Editor. We are going to address this in a service pack. Check the Hentzenwerke website for updates.

Figure 15. *The G2 Task List Editor provides some functionality not available with the native Visual FoxPro Task List..*

Future enhancements to this tool include capabilities to pack the metadata tables, filtering specific task types, recalling deleted tasks, copying a list of tasks to the clipboard (so they can be copied to an email, and generating some paper reports.

Menu Designer Hack Tool

http://hentzenwerke.com (once *1002 Things You Wanted To Know About Extending VFP* is released)
http://geeksandgurus.com
http://rickschummer.com

The Visual FoxPro Menu Designer is fundamentally the same old Menu Designer that we have been working with for years, since the days of FoxPro 2.6. There are a number of issues that have not been addressed that have pushed the author over the edge to create the first attempt at replacing the Menu Designer. The result of this endeavor is something we are calling the G2 Menu Hack form.

There are a number of issues that the tool attempts to address. The first is the woefully small area a developer is given to enter in the menu prompt and the results or action of the menu option. The text boxes provided on the Menu Designer are way too small. The other major frustration addressed is the need to open up a modal form for the Options. Jumping in and out of the Options dialog is a painful experience when you are attempting to view all the options to make sure they are correct before a build. The fact that the Menu Designer only shows ten menu bars to start is limiting when the designer is not resizable.

Figure 16. *The Visual FoxPro Menu Designer is showing its age in a state-of-the-art development environment.*

A couple of other things that makes the Visual FoxPro Menu Designer difficult to use is the way you navigate from one level to another. The Menu Level combobox is not exactly a friendly interface. The last thing we dislike (like there have not been enough already) is the menu option for the Menu Designer are scatter across two menu pads. The Menu pad is obvious enough to gain access to the Quick Menu, the adding and deleting of menu bars, the menu preview as well as the MPR generation process. You also have to pay attention to the View pad to gain access to menu's General Options and the Menu Options.

One word of caution when working with this tool and other Visual FoxPro metadata tools, make sure you make a backup of your metadata before hacking it. If you change something in the metadata and that change is not supported by the Menu Designer or the GenMenu.prg you can simply disable the menu and the ability to edit it in the native tools. This is the source code to your applications; safeguard it before hacking into it. The author takes no responsibility for your hacking actions.

The intent of the G2 Menu Hack is not to generate a new Menu Designer from the ground up. The original specifications dictate that it is 100% compatible with the native Visual FoxPro metadata. The reason for this is that Visual FoxPro developers still want to be able to use the native Visual FoxPro tools and as noted later in this section, the tool does not completely replace all the functionality of the native designer. It is capable of editing regular menus, shortcut menus, and top-level form menus.

The first benefit of the G2 Menu Hack is the ability to see and edit all the menu bar information on one page. The fundamentals page shows menu bar information for each menu item. If you were to look at the menu metadata file (MNX) you will see that the fundamental page shows records from record 3 to the end of the table. Also note that the treeview shows menu pad and bar items and there are multiple records in the menu metadata for menu pad items so as you navigate through the records there will be records on the fundamentals page that are not reflected in the treeview. The general page exposes records one and two of the menu metadata. The primary focus of this page is to expose the setup and cleanup code as well as the procedure code. Toggling between the fundamental and general pages will restrict the records that are in scope since they address different aspects of the menu metadata. The current record number of the metadata displayed is shown at the bottom of the form.

Figure 17. *The G2 Menu Hack tool is the first step at creating a replacement for the Visual FoxPro Menu Designer.*

To address the restrictive navigation of the Menu Designer, the G2 Hack Menu provides both a treeview and navigation buttons. The treeview allows you to quickly drill down the menu tree just like you would cascade the menu to pick the option you are accessing. The navigation buttons provide the ability to traverse the records in the menu metadata. As you move to a new record, the treeview is expanded to expose the record you are on in the metadata.

If you want to look up a specific record you can use the search (binoculars) and search again (binoculars with plus sign) buttons on the form toolbar. The search feature will present a dialog (see Figure 18) that allows you to select the column of the metadata to search in and a textbox for the text to search. If a record is found it is displayed in the form, otherwise a message is displayed indicating the failure to find the text. One caution when searching the prompt field is to make sure that you include the hotkey indicator in the text that you are searching. The text comparison used in the search is made using the $ operator and is case-insensitive. This means that searching the prompt column for"\<View" will locate "Pre\<view" and "\<View" menu bars. This feature uses the LOCATE (search) and CONTINUE (search again) commands so each time you use the search functionality it will find the first occurrence of the text.

Figure 18. *The G2 Hack Menu search dialog lets you pick the column to search and the text to be searched.*

The command window option provides the developer with the ultimate hack tool. It executes a single command (future implementations will allow full programs to be run). If you are use to doing mass replacements of **SKIP FOR** conditions via a **REPLACE ALL** command you can perform these within the tool and with the advantage of being able to revert to the original settings by not saving the changes.

This brings up the next feature, which is the changes are fully revertible because the tool implements buffering of the metadata. The changes are not saved until you press the save button on the toolbar, or close the form and respond yes to the question about saving your changes.

If you are comfortable with the technique of browsing the metadata, but like the user interface of the G2 Hack Menu, yet find a limitation of the interface, you can still browse the metadata within the tool. Again, this technique is safer than just browsing the metadata because it is buffered and you can reverse your changes.

The Visual FoxPro Menu Designer does a darn good job of keeping logically deleted records from hanging around. We still provide a **PACK** command just in case you find memo or record bloat in the metadata.

Finally, the G2 Hack Menu supports both Visual FoxPro 7 and menus built in previous versions of Visual FoxPro. The metadata layout changed with Visual FoxPro 7 to accommodate the new icon feature on the menu. If you are editing a menu created and edited prior to Visual FoxPro 7, the icon functionality is made invisible.

There are a few features in the Visual FoxPro Menu Designer that have not made it into this cut of the G2 Menu Hack tool. The first is that you cannot add/delete menu pads or bars. The tool does not provide a call out to the MPR generation, nor is does it have a menu preview mode, although the treeview provides the basic visual representation. It has no mechanism to add Visual FoxPro bar resources either. Another feature on the enhancement list is to provide a mechanism to open up the various code editboxes into the program editor to show code colorization and more importantly, allow the power of IntelliSense to be available to the developer. One other enhancement on the list is to make the tool resizable. Check the various websites on the About page for future updates of this evolving developer tool.

MsgSvcUI

http://stevedingle.com

Steve Dingle has written a user interface to Steven Black's public domain Message Services (MsgSvc) routine. The Message Services routine is a way to data drive all the messages from your application. You provide the conversion routine a key and it will display the message in a standard VFP MessageBox() style. Data driving the messages allows for language translations and keeps messages consistent in applications. It also allows you to change messages without changing the application executable.

Figure 19. *The MsgSvcUI application in action.*

Another nice feature is that you can preview the message interactively. Make some changes, test again, and repeat the cycle until you are satisfied. The report can be a little long, but generating a report is a great way to have your user review all the messages in the application without the need to run it.

FoxBox

http://rickschummer.com

FoxBox is a utility developed by Alex Korot of West Bloomfield, Michigan. It is not something you will use everyday interactively within the VFP development environment, but it could be a tool that you will find indispensable for your customers when you are onsite or doing some remote support via pcAnywhere. FoxBox is a Command Window (and much more) without having a full copy of VFP loaded on the customer site. It does require the same runtimes that are installed for your application.

This tool allows you to run any command in VFP that does not fire the “Feature not available” error. Therefore you can open up a table and browse it, you can perform SQL-Selects to inspect data, you can fire up a REPORT FORM, or recreate an index, or you can open text files. All the things you want to perform with VFP that you do not want to build into your executable can be accomplished via FoxBox.

Figure 20. *FoxBox is indispensable when debugging support issues on a customer's computer that does not have Visual FoxPro installed.*

The cursor browser (via browse button on FoxBox Console) displays a smarter browser. You can sort the cursor based on available indexes and by clicking on the column headers. The Structure button displays much more than the LIST STRUCTURE command produces.

RAS DBC ShowPlan Analyzer

http://geeksandgurus.com
http://rickschummer.com

This program is designed to be used by VFP 7 and documents the ShowPlan for all local views in the selected database container, using **`SYS(3054)`**. Each view in the database container is

opened, the ShowPlan is saved to a memory variable (one reason it requires VFP 7), combined with the others and eventually saved to a file for human analysis. The results are displayed after the view performance is checked.

The optional Show SQL (third parameter) might seem redundant to the 2 and 12 options on ShowPlan Level for **`SYS(3054)`**, but it is the author's experience that these do not work as expected. The utility also does some basic formatting of the SQL to make it wrap and be more readable.

The database path is temporarily added to the SET PATH so developers who have separate view database containers from the table database container (and have them reside in the same directory) will have automatic access to the table database container. If the table database container resides in a separate directory, add the database directory to the SET PATH before running this utility or you will be prompted and have to select it.

All this is handled in a private datasession so the environment is safely re-established when completed.

List Structure

The LIST STRUCTURE command has been around since the beginning of Xbase. The problem with the current rendition in VFP is that it is nearly useless if you are using long field names since it truncates them in output.

This particular pain has triggered plenty of VFP developers to develop their own little LIST STRUCTURE utility. You can surf the web and find them all over the place. I found 4 different ones on the UniversalThread.com, six more on CompuServe's MSDevApps forum.

Sources to Grab Free Tools

Visual FoxPro

http://msdn.microsoft.com/vfoxpro

All the built in developer tools that ship with Visual FoxPro including the Class Browser, Object Browser, Coverage Profiler, Task List, Wizards, and Builders are included in a file called XSource.zip, located in the Tools\XSource directory under the VFP home directory. The product is not free, but the source code was not available until VFP 6.

The MSDN site has numerous code examples, updates to the various VFP packaged tools, and whitepapers with help and tools available.

CompuServe MSDevApps Forum

http://forums.compuserve.com/vlforums/default.asp?SRV=MSDevApps

This forum and the predecessors before it on CompuServe have serviced the FoxPro community for over a decade. There is source code to developer tools going back to the FoxPro 2.x days. There are nearly 200 files in the VFP Utilities/Tools section, over 2000 for FoxPro 2.x.

Universal Thread

http://universalthread.com

The Universal Thread is the forum recognized by Microsoft as the official support forum for Visual FoxPro. This forum, while not active as long as CompuServe, serves as the home to more than 700 downloads for Visual FoxPro.

FoxWiki

http://fox.wikis.com

The FoxWiki does not have a download section, but that does not mean that there are not programs galore available to use as tools in development. There is tons of code posted on this website that I have used in my developer tools. This is the single best knowledgebase of VFP information available.

Virtual FoxPro User Group

http://www.vfug.org/

The Virtual FoxPro User Group is a nice site with information and downloads available. Costs nothing to join and has a great newsletter to boot. Almost 200 FoxPro files available for download.

Vendor sites

The vendors always have commercial products, books, or training to sell, but many of these sites offer free add-ons or tools that their customer base has written that extend the commercial products. It is worth checking into any of the sites that offer VFP products to see if they offer any freebies or add-ons. They often have links to other VFP sites as well.

- http://www.eps-software.com/ (EPS Software)
- http://www.f1tech.com/ (F1 Technologies Visual FoxExpress)
- http://www.hentzenwerke.com/" (Developer's Studio, Hentzenwerke)
- http://www.oakleafsd.com/ (Oakleaf's The Mere Mortals Framework)
- http://www.strategicedge.com/index.htm (Strategic Edge's FoxSpell spell checker)
- http://www.stonefield.com (Stonefield Systems Group)
- http://www.takenote.com/ (TakeNote Technologies)
- http://www.visualmaxframe.com (Vision Data's Visual MaxFrame Professional)
- http://www.west-wind.com (West Wind Technologies)

Periodicals

The various periodicals that service our industry cover a variety of topics. From time to time they publish reviews of commercial developer tools as well as some of the "grab" tools from fellow developers.

- http://www.pinpub.com/foxtalk (Pinnacle Publishing's FoxTalk)
- http://www.advisor.com/wHome.nsf/wPages/FAmain (Advisor's FoxPro Advisor)
- http://www.code-magazine.com (CoDe Magazine)

Individual Developer and User Group Websites

Individual developer sites are always showing up. The best way to find developer websites are in the message forums. Developer signatures of developers with a website usually sign all their messages with a link to their pages. I find some of the best stuff at these sites. They are always worth a look.

- http://rickschummer.com (shameless self promotion of my website)
- http://www.craigberntson.com (Craig Berntson's website)
- http://www.jamesbooth.com (Jim Booth Consulting)
- http://www.mikehelland.com (Mike Helland)
- http://www.ml-consult.demon.co.uk/ (Mike Lewis Consultants)
- http://www.jjtc.com/Fox/ (Neil's FoxPro Resources)
- http://www.stevenblack.com/ (Steven Black's website)
- http://www.stephensawyer.com (Sawyer's Lair)
- http://www.stephendingle.com (Stephen P. Dingle)
- http://www.tedroche.com (Ted Roche's website)

Why Buy?

There are plenty of other VFP developers who struggle the same way you are and may have written tools first. (sound familiar?). Developers have different pain thresholds, but we all work with the same great development tool so we all hit the same weaknesses and have the same set of strengths take advantage of in our development. A number of vendors have identified weaknesses with VFP and have created commercial products to ease a specific pain, or more importantly, enhanced how we approach our development.

The biggest advantage of buying a developer tools is the cost. Typical developer tools range from $50 to $500. This works out to less than a day's worth of billable time. Most of the simple developer tools that I have built have more than a couple of hours into them. They easily save me the time when used again and again, but if I can buy the same tool and work on billable work I am typically way ahead financially.

The second reason is that the vendors have a vested interest in the success of the product and work hard at providing updates and new functionality to entice you into upgrading and purchasing for the first time. Continuous improvement will give developers better productivity and improved development techniques, which should lead to more solid code and better solutions for our customers.

The third reason to buy is that the time already invested working out the bugs. Commercial products have an additional advantage in the fact that they have a base of developers already using the product. These developers have different needs for the products to support and offer enhancement requests and

What are the real differences between "Grab" and "Buy" tools? Sometimes there is very little difference; sometimes the differences are pretty substantial. The big difference is that the commercial packages usually offer more features, have a commercial polish, and come with help files or developer documentation. The freebies are usually offered to the community "as is". This does not make them less valuable, but they are free, usually feature complete to the original developer's needs, and offer little in the way of help.

Tools to Consider Buying

Stonefield Database Toolkit (SDT)

Stonefield Systems Group
1112 Winnipeg Street
Suite 200
Regina, SK Canada
S4R 1J6

Phone: 1-800-563-1119
Fax: 306-586-5080
http://www.stonefield.com/
sales@stonefield.com

SDT 6.0 is available for US$349, which includes a one-year subscription for updates and support. One year subscriptions run US$129 for a single license, US$449 for a 5-user license, and US$669 for a 10-user license.

Stonefield Database Toolkit is the single most important tool I have used in developing applications with VFP databases and tables. Why? The reasons are numerous.

Figure 21. *Stonefield Database Toolkit in action with the Database Explorer and the Extended Table Designer.*

Reason #1: Field changes made to a table in the SDT Extended Table Designer do not break views. Changes made to tables in the native VFP designers can break views. The only way to fix them is to recreate them. If you manage views programmatically this is not as much of a problem, but if you are not then you are likely to get the "Base table fields have been changed and no longer match view fields. View field properties cannot be set (Error 1542)" from VFP when you try to open or edit the view. The help on this error should plainly state, "to avoid this error in the future, go out and buy a subscription to Stonefield Database Toolkit".

Reason #2: Reindex() method. Recreating index routines are needed when you have VFP data. Periodically, for reasons usually beyond your control like a power outage, or hardware failure, or the user just flipping the power button, cached indexes on the client do not get written to the CDX file associated with the table. The indexes get corrupted (out of sync with the table records) and the best way to fix them is to do a `DELETE TAG ALL` and rebuild each tag. While it is easy to write code to do this, the pain is in the maintenance as new tags are added and obsolete tags

are removed. Developers can also write code to sweep through a data dictionary that they create, at the cost of hours to develop. Doug and the crew at Stonefield have already written a metadata driven routine that just plain works. There is no need to keep track of the indexes, all you need to do is Validate() the metadata before shipping. There is another routine that will check to see if the indexes need to be updated.

Reason #3: Update() method. Like the Reindex() method, SDT includes a routine that will automate implementing changes made to the data model in development to a production database. A developer can easily forget a change made in development and watch as the program starts failing on field not found errors. It will change field sizes, rename field names, change the data type, and any thing else that `ALTER TABLE` can perform, all in an automated process.

Reason #4: Field Repository. Have fields that are common to different tables like a user id and time stamp field and want to make sure they are defined the same in all tables? Have different applications that maintain addresses and want them consistent? The Field Repository allows you to define fields and add them (as well as update them later) base on one definition.

Reason #5: Better table designer interface than the VFP tools. This may be more an editorial reason than the others, but I really prefer the interface Doug presents than VFP's. The only thing VFP does better is the initial index creation, because VFP is more forgiving, it will fail on troubled indexes and create the rest, SDT fails on one and the rest are not created.

Reason #6: Integrates with Steve Sawyer's ViewEditor. I now exclusively create views with Steve's editor, and it rocks. Doug has added a switch to SDT that will use ViewEditor if you indicate that the view is not to be opened in the VFP View Designer.

Overall I cannot say enough about this tool. The documentation is some of the best I have seen for third-party developer tools. If you find something not working as advertised I have seen fixes sent in an email response to my issue posted. The Update page on the Stonefield website gets regular visits because the Stonefield folks are pretty consistent at getting changes posted that are needed by their customers.

FoxAudit

TakeNote Technologies
P.O. Box 99271
Raleigh, NC 27624
(919) 870-9000 - phone
(919) 845-7666 - fax

Phone: 1-800-563-1119
Fax: 306-586-5080
http://www.takenote.com
cniles@takenote.com

FoxAudit 6.0 is available for US $259, with site licenses available. It ships complete with source code, documentation, royalty-free distribution rights, and a 30-day money back guarantee.

FoxAudit 6.0 is a Visual FoxPro 6.0 class that provides complete audit trail support to Visual FoxPro 6.0 DBC-based applications. It empowers Visual FoxPro developers by allowing them to add complete, automatic, client/server-like, audit trail support to the Visual FoxPro database container. It works by calling a stored procedure from the Insert, Update and Delete triggers of a Visual FoxPro table. If a record gets updated, inserted or deleted, FoxAudit knows it and will log the update to the transaction log table.

Frameworks

Visual FoxExpress (VFE) **F1 Technologies** 335 North Superior Street Toledo, Ohio 43604-1427 Desktop, Client Server, Web	Phone: 419-255-6366 Fax: 419-255-6371 http://f1tech.com info@f1tech.com	VFE 6.3 is available for US$699, which includes a six month subscription for updates and online support. One year subscriptions run US$349 for a single license, multi-developer discounts are available.
Visual MaxFrame Professional (VMP) **Vision Data Solutions. Inc.** 17501 East Hwy 40 Independence, MO 64055 Desktop, Client Server	Phone: 1-888-904-7900 Fax: 816-373-3020 http://www.visionds.com/VMPSite/ sales@visionds.com	VMP is available for US$499, which includes a one-year subscription for updates and online support. One year subscriptions run US$199 for a single license, multi-developer discounts are available.
The Mere Mortals Framework **Oak Leaf Enterprises** 952 Rockledge Drive Charlottesville, VA 22903 Desktop, Client Server, Web	Phone: 434-979-2417 Fax: 434-979-5637 http://www.oakleafsd.com/ oakleaf@oakleafsd.com	Mere Mortals is available for US$399. Your purchase includes 30 Days of Free *direct e-mail* support that doesn't start until you ask your first question.
Web Connection (WC) **West Wind Technologies** 32 Kaiea Place Paia, Hawaii 96779 Web	Phone: (808) 579-8342 Fax: (808) 579-8547 http://www.west-wind.com/ rstrahl@west-wind.com	Web Connection 3.x is available for US$399. Support provided on the West Wind Message Board.
Active FoxPro Pages (AFP) **AFPWeb Dot Com, Inc.** 2400 Falkner Rd Orlando, FL 32810 USA Web	Phone: No contact phone in USA Fax: http://www.active-foxpro-pages.com/ OR http://www.afpweb.com/ Sales@AFPWeb.Com	AFP is available for US$499.

Some developers just want to write their own framework, and I encourage you to do so. Personally, I have written three of my own (FPD, FPW & VFP). It is a terrific way to better understand what you like and dislike in a framework and it is a good way to learn VFP. Developing your own framework allows you to customize it to your specific needs. Commercial frameworks are designed to provide developers with generic functionality, including functionality you may not need. They also provide code to protect developers from their own coding mistakes and layers of defensive code.

Today I use a commercial framework for my day-to-day development projects. I am very familiar with two major frameworks. Why am I using a commercial framework after developing my own for VFP? One, it takes time to architect, develop (six months minimum, even for experienced developers) and perfect. Two, I have to maintain it. Three, it was not as robust as

the commercial frameworks. I am in the business of developing custom solutions for my customers, not maintaining the corporate framework. I might change my mind if my company grows to 15 developers and our project base is diverse.

Selecting a framework is as individual a process as there is in the software development industry. The things that determine the selection of a framework include coding style, desired application architecture (no-tier, 2 tier, n-tier, desktop, client/server, or web), wizards or built in wizardry, your experience, industry you are developing for, and plenty of others. There is no way I can write up the various frameworks available, but there are a number of excellent write-ups and comparisons on the FoxWiki (http://fox.wikis.com/wc.dll?Wiki~FrameworkFeatureChart~VFP). Pinnacle Publishing's FoxTalk also presented several articles reviewing the pros and cons of the various application frameworks. These articles are available online (http://www.pinpub.com/foxtalk) if you are a subscriber. If you are not, $129 for a year might be worth it just to review these articles on a most important topic.

xCase Professional

RESolution Ltd. 641 Lexington Avenue Suite 24/SAI New York, NY 10022	Phone: 1-800-283-8957 Fax: 212-688-1327 http://www.xcase.com/ info@xCase.com	xCase Professional is available for US$799, xCase For Fox is available for US$399, xCase Viewer is available for US$149.

Ever been frustrated with the VFP Database Designer? Have experience in a CASE tool that does not generate VFP databases? Have products that are developed to support both VFP and SQL-Server databases? Ever wanted to print out the database design as it appears in the design surface?

Figure 22. *View of the VFP sample TasTrade database in xCase.*

xCase is a tool used to interactively work with the database design and perform maintenance on the database design. In addition to creating tables and setting up relations and the typical tedious work interacting with the setup and ongoing design, xCase Professional will generate the database in a variety of target database products such as VFP, SQL Server, Oracle, Sybase, SQL Anywhere, Informix, DB2, Interbase, and Jet (VB and Access). You can create difference views of the tables and the relationships as well. xCase can also reverse engineer and existing database model by reading the database container. Output of the data model includes printing out to a Windows' printer in case you want to generate some wallpaper for the office.

The folks at RESolution also have a product called the xCase Viewer that is distributable to the customers so they can review the database design without needing a full version of xCase.

West Winds HTML Help Builder

West Wind Technologies 32 Kaiea Place Paia, Hawaii 96779	Phone: (808) 579-8342 Fax: (808) 579-8547 http://www.west-wind.com/ rstrahl@west-wind.com	HTML Help Builder is available for US$199. Support provided on the West Wind Message Board.

I don't think I'm the only developer that finds writing help for an application painful. It is not the content that is painful, but working with the HTML Help Workshop provided with Visual Studio. The HTML Help Builder has been around for a few years and basically removes the pain of building HTML Help files.

Figure 23. *West Winds HTML Help Builder Interactive Development Environment in action.*

HTML Help Builder will ease you through the steps to build the CHM file once you enter in the content. It does all the work of interacting with HTML Help Workshop via automation and shelling out. It has a built in editor and allows you to have VFP code evaluated as the help file is being generated. All the source is saved in a DBF file and there is a COM interface if you want to write code to automate the process of generating the content.

This product will also help build technical documentation because it will import the comment text from an existing Visual Class Library. A number of third party VFP tools have utilized this tool in building their help files.

Recover

Abri Technologies 126 Blue Jay Rd. Pierre, SD 57501 USA	Phone: 605-224-0660 Fax: None http://www.abri.com/ Email to sale available via website so they avoid getting SPAM	Recover is available for US$70 for a single user license, US$250 for royalty-free license (redistributable with applications).

The dreaded phone call comes in from the client. They had the ultimate a power failure last night during a lightning storm, even the reliable battery backup power supply failed and the server went down during the nightly batch update process. The power has been restored by the utility company, and the hardware suffered no ill effects. The applications was started back up, but they are experiencing “not a table” errors and some tables no longer have the memo fields displaying correct information. What can you do? While DBF and FTP files are stable during day-to-day operations, occasional glitches do happen.

This is where a product like Recover comes to the rescue. A quote direct from their website:

> “Recover goes far beyond simple header repairs and can pick out records and/or memos from ‘garbage’ files without user interaction using the proprietory Trace (patented), SPM and LFS techniques.”

The product is easy to use, performs the needed diagnostics on the troubled tables and will fix them without the need of user interaction if desired. There is also an interface so you can interactively work to fix the affected tables if you like.

DynaZip

Inner Media, Inc. 60 Plain Road Hollis, NH 03049, U.S.A	Phone: 800-962-2949 Fax: 603-465-7195 http://www.dynazip.com/ sales@innermedia.com	DynaZip – Complete is available for US$299, DynaZip – AX is available for US$149, DynaZip – GT is available for US$249.

Most computer users have run across a compressed file in the ZIP format. DynaZip is an ActiveX control that provides developers a COM interface to generate ZIP files, add files into a compressed file, extract files from an extracted file (to memory or disk), and get properties of the file like file count and comments.

Example of UnZipping files from a compressed file:

```
with thisform.oUnZip
   * Setup up the action
   .ZipFile                = 'ZIPDIVWASTE.zip'
   .Destination            = lcDestination
   .FileSpec               = '*.*'
   .ExtProgTitle           = "Restoration from data backup"
   .BackgroundProcessFlag  = .F.
   .AllQuiet               = .F.
   .NoDirectoryNamesFlag   = .F.
   .RecurseFlag            = .T.
   .OverwriteFlag          = .T.

   * Actually perform the Add of the files
   *  0 - No Action
   *  1 - Count All Zip Members
   *  2 - Get Next Zip Information
   *  3 - Count Named Zip Members
   *  4 - Get Next Named Zip Information
   *  5 - Get Comment Size
   *  6 - Get Comment
   *  7 - Get Indexed Zip Information
   *  8 - Extract
   *  9 - File To Memory
   * 10 - Memory T Memory
   * 11 - File to Memory Stream

   .ActionDZ = 8

   * Note error code set to zero to see if one is set during operation
   thisform.lblErrorCode.Caption = alltrim(str(.ErrorCode))
   thisform.lblErrorCode.Visible = .T.
endwith

if thisform.lblErrorCode.Caption > '0'
   thisform.lblErrorCode.Caption = "Trouble decompressing the files you requested, " +  "
                                   "see the error code for details."
   thisform.lblErrorCode.Visible = .T.
else
   thisform.lblErrorCode.Caption = "I'm done decompressing " + ;
                                   alltrim(str(thisform.oUnZip.ReturnCount)) + ;
                                   " files you requested from " +
                                   lower(alltrim(thisform.oUnZip.ZipFile)) + ;
                                   "to the " + lower(alltrim(thisform.oUnZip.Destination))
   thisform.lblErrorCode.Visible = .T.
endif
```

Issues with the DynaZip are hard to find. The control works well with Visual FoxPro and

MSDN Universal Subscription

Microsoft Corporation 1 Microsoft Way Redmond WA 98052-6399	Phone: 800-426-9400 Fax: None http://msdn.microsoft.com/subscriptions/	MSDN Universal is available for US$2,499 for a single license, US$1,999 annual renewal. Discounts available to registered owners of Visual Studio products.

The MSDN Subscription is a must have package from Microsoft. Every developer tool Microsoft has currently available for public consumption is included on CD format (and available for download from the subscriber website).

The package includes all of the Microsoft Operating Systems (all supported versions, in all languages, including BackOffice), all the developer tools (Visual Studio and VFP), all the various packages of MS Office, MSDN Library (help files and KnowledgeBase), and productivity tools (like Visio, Visual SourceSafe, and Project). This includes the latest released versions, updated Service Packs and beta versions of upcoming releases. Microsoft also throws in the betas for all the products and OSes as well. If you buy Visual Studio Enterprise, you are already half way to owning a copy of MSDN Universal and you get so much more value.

If the Universal level is a bit pricy, you should know in October 2001, Microsoft announced that there are now five levels of MSDN. The Professional level will get you a copy of Visual FoxPro, all the other developer tools, and the operating systems. This is a bargin at $900.

Go to http://msdn.microsoft.com/subscriptions/prodinfo/levels.asp for a complete listing of what is included in the subscriptions.

Conclusion

I sincerely hope that you found useful information on how to start developing tools that enhance your ability to kick out the finest custom applications known to mankind. I hope you found some inspiration in the information presented in this session. Most of all, I hope you decide that developing tools for VFP is both beneficial to yourself and the fine community that we all belong to. I encourage you to share the fruits of your labor with the Fox Community so all developers get a chance to benefit the same way as you are. Thanks for taking the time to read these session notes.

Special Thanks

I want to thank Patty Nowak, who will always be my first editor, for taking the time to read over these extensive notes and for the numerous corrections she suggested. Time after time she has made me look better as a writer than I deserve.

To all the developers who have inspired me to write developer tools and who offer some of the best darn tools around. A special thanks to Doug Hennig, Mike and Toni Feltman, Steve Sawyer, Ken Levy, Drew Speedie, and Steven Black.

The Fox developer community for their feedback on the works I have offered.

Copyright

Copyright © 2001-2002, Richard A. Schummer. All Rights Reserved

Author Profile

Rick Schummer is a partner at Geeks and Gurus, Inc, a full service computer solutions provider in Detroit Michigan. After hours he enjoys writing developer tools that improve his company's productivity and occasionally pens articles for his favorite Fox periodicals and user group newsletters. Rick is a co-author of 1002 Things You Always Wanted to Know About Extending Visual FoxPro and 1001 Things You Always Wanted to Know About Visual FoxPro, a founding member and Secretary of the Detroit Area Fox User Group (DAFUG). He is a regular presenter for these organizations, other user groups, and at GLGDW 2000, 2001, and EssentialFox 2002. He can be reached at raschummer@geeksandgurus.com and http://www.rickschummer.com