Visual SourceSafe

Session Number

Ted Roche
Ted Roche & Associates, LLC
278 Kearsarge Avenue
Contoocook, NH 03229-3103
Voice: (603) 746-5670
Web: http://www.tedroche.com
Email: tedroche@tedroche.com

Overview

Every developer, whether working singly or as part of a team, should be using Visual SourceSafe to keep their code safe, create a history of changes, and act as a backup and "Grand Undo." In this session, Ted presents tips on installation, configuration, and needed maintenance to make SourceSafe work flawlessly for you.

Introduction: Why use SourceSafe?

Visual SourceSafe is a version control system that allows you to store files and the changes to those files over time, to retrieve a file version from a point in time, and to report and analyze the changes to a file over time. So what, you say? Well, many developers have had the experience of compiling their application, only to find a problem they are sure that they fixed reappear. With SourceSafe, the ability to retrieve previous versions gives you a "Grand Undo" function to reverse or retrieve earlier versions of a file.

SourceSafe also works as a great repository for items you don't wish to misplace. Workstations come and go, developers pass through many jobs, but the SourceSafe database, if properly backed up and cared for, can last a long time. This can be handy if you need to support clients with software several years old, or need to retrieve documents more than a few months old.

In a team environment, with multiple developers working on shared source code, Visual SourceSafe is essential to ensuring that changes are not lost and that developers are all working on the most recent version of code.

Installation: What goes where

Before you can start working with Visual SourceSafe, you need to get it installed. Installation is not difficult, but you'll want to make sure that files end up in the right place. To do this, of course, you need to know where the right place is. First, we'll look at how SourceSafe works, and then perform the installation.

Visual SourceSafe is file-server based

Visual SourceSafe is a file-server based architecture, in contrast to a client-server architecture. In a file-server model, all of the executables (the "clients") are run on their own workstations, and make read and write requests to the shared files. There is no server executable running, only a machine with files to share. Visual FoxPro works exactly the same way, where the VFP executable runs on local workstations and the data can be shared on a network file server. No portion of VFP is running on the file server, it is just a dumb beast serving file requests. The difference between this and the client-server model is that in the client server model, the clients make requests directly to another executable, the server. It is this server that actually does the reading and writing, and this server can have a set of rules it follows to ensure the security, integrity and proper performance of the application. A file-server model can be less reliable, since a single client can directly damage the data store.

Executables on server for convenience

Understanding that Visual SourceSafe is a filer-server model can help to make the installation process clearer. When you first start the Visual SourceSafe installation, you will be asked whether you want to perform a Client Install, a Server install, or a Standalone Client install. Confusing as that sounds, here's what it means:

- a Client installation only installs the executables onto a local workstation
- a Server install is meant to be performed on the file server, in order to set up the data structures needed for the shared Visual SourceSafe database
- the oxymoronically-named Standalone Server install is a combination of the two, intended for standalone workstations or laptops where the data structures and the executables all are on the same machine.

The setup program itself is unsurprising. It asks for locations to install the software. Typically, I prefer to install the SourceSafe database (the Server install) in a location that can be accessed by all developers, often off the root of a file server volume, and shared directly as the S: drive (this isn't required; it just makes it easier for me to remember: S is for SourceSafe). The client software is usually installed in the Program Files directory, although you may choose whatever location you prefer.

A problem with this model occurs when the inevitable service pack appears. Service Packs are designed to update the configuration on individual machines, and not on both server and client installs. When the Service Pack is run on the server machines, only the binary files are updated; the setup files for the client install still contain the original files.[1] Hence, clients need to have the Service Pack run on them as well. Each future installation of the client software from the server will also require that the Service Pack be run on the target machine after the installation. In past versions of SourceSafe, Microsoft has provided the instructions to manually update the client installation files to the latest service pack, but unless a large number of installations must be performed, this is both more time-consuming and difficult that just running the Service Pack on each client machine.

---

[1] See Microsoft Knowledgebase article # Q234526, "PRB: Visual SourceSafe 6.0 SP3 Netsetup Installation Problem"

Configuration

Security

Currently, Visual SourceSafe maintains its own user list and, if enabled, individual rights can be granted or revoked on individual projects. While most small development shops will not want to add the administrative burden of maintaining this feature, larger shops often find it necessary, and an onerous task to maintain. It is hoped in future versions that integration with the Windows security model, with the ability to assign group rights, will make this task much easier.

Since Visual SourceSafe maintains source code as separate files on disk, and since anyone with permission to use SourceSafe must have complete rights to the data directories, login and security are really only an inconvenience to someone determined to access source code. If tighter security is required, setting up separate databases and using the network rights mechanism is required to prevent unauthorized access to source code.

Multiple checkouts

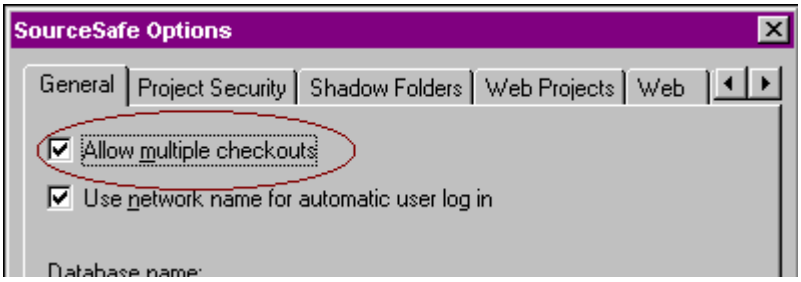

Figure 1: Multiple checkouts should be turned on.

Maintenance

Analyze

Analyze is an essential utility for analyzing the health of a VSS databases, detecting inconsistencies and errors, repairing some of the more simple errors, and compacting unused space in the database. **ANALYZE should be run periodically (at least weekly) on production databases.** The results of Analyze can be a little difficult to decipher, but the Administrator help file (and also Microsoft KnowledgeBase article Q152807) list the more common errors and documents fixes, where available, for these errors. After making a backup of the database and attempting to fix it, always run Analyze again to verify that the problem has been fixed, and to find out if fixing this error brought forth any other errors the first problem may have been masking. In many cases, the Analyze, Backup, Fix cycle may need to be run several times.

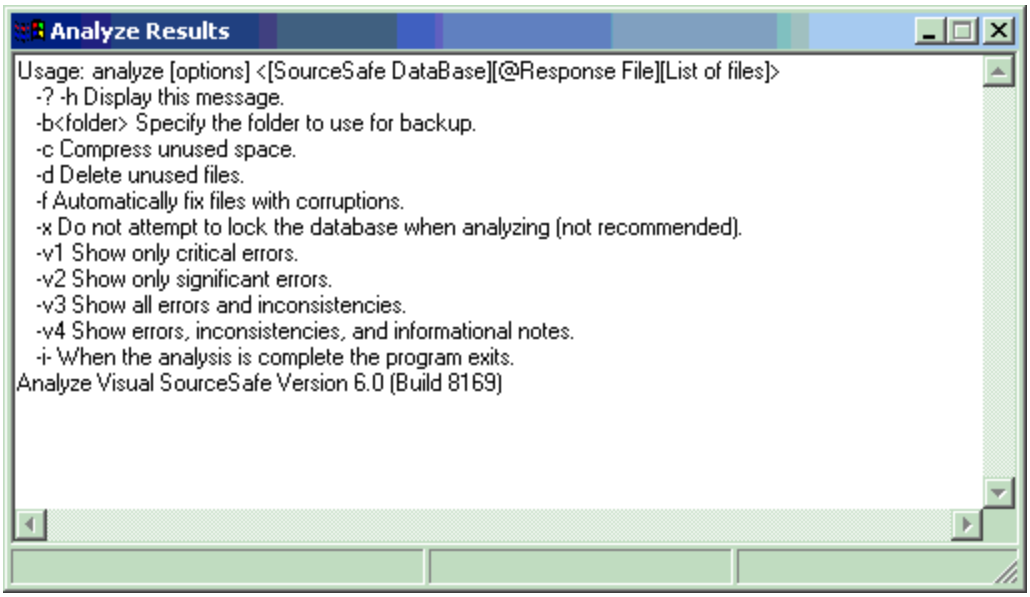

Figure 2: ANALYZE -H will prompt you with options

With the tools available within Windows NT or Windows 2000, it is not difficult to set up a scheduled task to run the Analyze program on a regular basis, and send the results to a text file. From there, you can process the file into your email system, compare it against previous values, or parse it in FoxPro to determine if actions are required.

Physical File List

Visual SourceSafe stores the entire tree of source code, as well as the tree structure itself, as a large number of files. The files are generated with a sequential series of alphabetic names. There are two files with each name, one with no extension, and the second with a ".A" or ".B" file extension. The file with no extension stores the history of changes to that node in the tree. The file with the extension is the actual contents of that node. If the node is a folder, it contains the folder name and pointers to the files contained in that node. If the node is an actual file (a leaf node), the file with an extension contains the source code itself. Each time the file is changed, a new copy of the file is written out, with whichever extension is available. If the write is completed successfully, the original file is deleted. So, knowing how these obscurely named files match up to the files in your project can give you the ability to recover your source code even if the database was to become severely corrupted.
The format for generating a Physical File List is:

```
<PATH>\SS PHYSICAL $/ -R -O@Physical.Txt
```

Where <PATH>\SS points to the SS.EXE file (installed in the Win32 subdirectory of the SourceSafe install), PHYSICAL is the keyword that tells SourceSafe what to do, $/ tells SourceSafe to start from the root of the projects tree, -R to act recursively through the entire tree, and –O@ designates the name of the output file. A word of caution: this command appends to an existing file, if it exists. You'll want to delete the existing file if this is not the desired behavior.

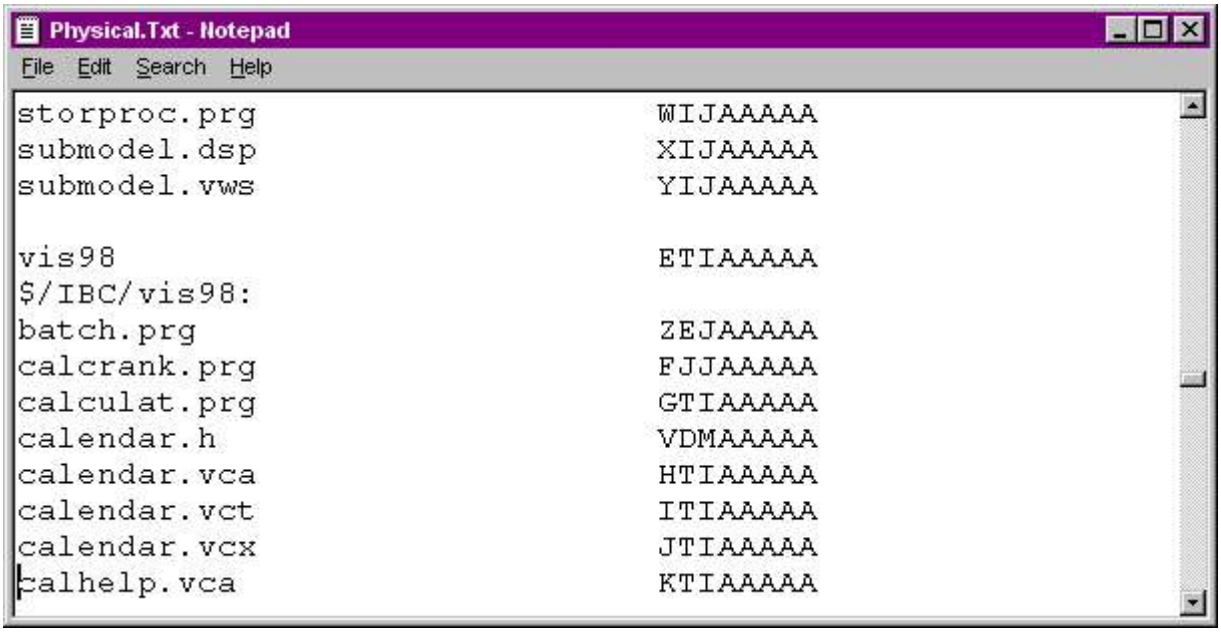

Figure 3: Results of a Physical File List

Visual SourceSafe lacks an interface to provide you with some key management information. For example, if you find that your database has grown significantly in size, how to do you determine what caused the size change? Using the physical file list, you can write programs in VFP to help you determine this information:

```
*************************************************************
      * Program....: PARSEPHY.PRG
      * Version....: 1.0
      * Author.....: Ted Roche
      * Date.......: June 7, 2000
      * Notice.....: Copyright © 2000 Ted Roche, All Rights Reserved.
      * Compiler...: Visual FoxPro 06.00.8492.00 for Windows
      * Abstract...: Parse the physical file list
      ************************************************************
      **

      * Create a working cursor

      create cursor curlist (mFilePath M, ;
                             cFileName C(8), ;
                             iFileSize i, ;
                             iSrcSize i)
      index on cFilename tag cFilename

      local lcFileName, lcList, laList[1], lcLine, lnLine
      lnLine = 1

      lcFilename = "S:\Current\Physical.txt"
      IF NOT FILE(lcFileName)
        lcFileName = LOCFILE(lcFileName, "TXT", "File?")
      ENDIF

      * Low-level and array techniques proved difficult
```

```
* for this 2+ Mb file.
* MEMLINES() would appear to hang with > 64k lines

* lcList = FileToStr(lcFileName)
* ALINES() is limited to 64k lines - Current is ~75K
* lnLine = ALINES(laList, lcList, .T.)  && convert to an array

select 0
create cursor fred (cString C(254))
append from (lcFilename) type sdf

* First line is different from all others, so
* fake a record into the table
GO TOP
lcLine = ALLTRIM(Fred.cString)
* First line is the sourceSafe root: $\, usually aaaaaaaa
lcPath = "$\:"
insert into curList values ("Root", ALLTRIM(lcLine), 0, 0)

SCAN REST
  lcLine = ALLTRIM(Fred.cString)

  IF EMPTY(lcLine)  && skip empty lines
    LOOP
  ENDIF
  IF RIGHT(lcLine,1) = ":"  && it's a path
    lcPath = lcLine
  ELSE  && if it's not a path, it's a file
     insert into curlist values ;
       (lcPath + alltrim(LEFT(lcLine,32)), RIGHT(lcLine,8), 0,
0)
  ENDIF
ENDSCAN

* Now update the files for size
local laDir[1], lnCount, lcSubDir
lcPath = "S:\Current\Data\"
IF NOT DIRECTORY(lcPath)
  lcPath = GETDIR(lcPath, "Data?")
ENDIF

select curList

local lcFile
* Loop through each subdirectory, "A" through "Z"
FOR m.i = 0 to 25
   lcSubDir = lcPath + CHR(ASC("A")+ m.i) + "\*.*"
   lnCount = ADIR(laDir, lcSubDir)
   ? "Updating directory " + lcSubDir
   FOR m.j = 1 to lnCount
     lcFile = UPPER(laDir[m.j,1])
      * For each file, see if it exists in the cursor, and
```

```
      * if so, update the size of the file
      * LOCATE FOR cFileName = UPPER(laDir[m.j,1])
      * We use SEEK rather than UPDATE because we only
      * want to save the size once -
      * shared files are referred to more than once.
      IF SEEK(JUSTSTEM(lcFile))
        IF EMPTY(JUSTEXT(lcFile))  && it's a VSS delta file
          REPLACE iFileSize with laDir[m.j,2]
        ELSE  && it's source
          REPLACE iSrcSize with laDir[m.j,2]
        ENDIF
      ENDIF
   ENDFOR &&* m.j = 1 to lnCount

ENDFOR &&* m.i = 0 to 25

* Display the results
brow fields x=left(mFilePath,50), cFilename, iFileSize,
iSrcSize

*!*  sum iFileSize, iSrcSize, iFileSize + iSrcSize
*!*  * 1,637,080,265 by calculation here.

* Get top Groups with:
* Top groups are the nodes off the root path in the form
* of $/Project/subproj/subproj/subproj....

select PADR(LEFT(mFilePath,AT("/",mFilePath,2)),20) ;
                                    as cTopGroup, ;
       COUNT(*) ;
 from current ;
 having NOT EMPTY(cTopGroup) ;
 group by 1 ;
 order by 1 ;
 into cursor TopGroup

* Total the sizes for each group
select topgroup.cTopGroup, sum(iFileSize), SUM(iSrcSize),

SUM(iFileSize+iSrcSize) as iTotal ;
  from TopGroup inner join Current on ;
       Current.mFilePath = ALLTRIM(cTopGroup) ;
 group by 1 order by 3 DESC
```

Listing 1: Sample code to parse Physical File List

Moving Projects and Duplicating Databases

Microsoft's suggested technique for moving projects between different databases is to use the Archive and Restore options within the SourceSafe Administrator. The advantage of such a technique is that all history is preserved between databases. However, in many cases, the history is not required. It is far simpler to perform a "Get Latest Version" on all files from the source database and "Add" all files into the target database. Also, I have encountered problems with Archive and Restore reporting satisfactory performance, while Analyze reports numerous file

problems. This may have been caused by very old and poorly maintained databases so, if preserving the history is important to you, this avenue is worth a try.

Duplicating Databases

Since databases are a set of files on disk, it is tempting to duplicate a database's structure and content by simply copying the Data directory to a new location. This will work, however, there is a hidden danger in doing this, and a fix from Microsoft that compounds a problem into a catastrophe.

Each database is uniquely tagged with a GUID to distinguish it from all others. While VSS will work fine switching between databases with identical GUIDs, Backup and Restore will fail if an attempt is made to transfer files between two databases of identical GUIDs. The Restore program mistakenly thinks it is writing to the original database, and it can produce error messages and corruption of the target database.

Microsoft documents this problem in Microsoft Knowledge Base article Q176780, and provides links to an executable, GUIDSCAN.EXE that, it claims, allows the operator to reset the GUID, stored in the UM.DAT file, to make the databases unique. However, this program does not work in version 6.0 and has corrupted UM.DAT files on several installations on which I tried it. Fortunately, the precaution of making backups before attempting this operation has saved me from a costly rebuilding of my database. Microsoft has been aware of this issue since at least April of 2000, have admitted to me that it is a problem, but continue to have the executable on their website! Amazing.

Repair

Regular maintenance of Visual SourceSafe databases insure that, when repairs are needed, most errors can be fixed easily, or backups will minimize the amount of code lost. Microsoft's Knowledge Base contains an article (Q152807) that duplicates the information available within the Administrator Guide (part of SSUSEXP.CHM, installed as part of the client install) that lists common problems reported by Analyze and fixes for them, if possible.

To Integrate or Not to Integrate?

1. One of the bit attractions to using Visual SourceSafe with Visual FoxPro has been the promise of integrating the Visual SourceSafe functionality directly into the Visual FoxPro Project Manager. By modifying the Tools/Options dialog's Project tab (see Figure 5), the SourceSafe status of files and the ability to manipulate the files in SourceSafe is available directly within the Project Manager.

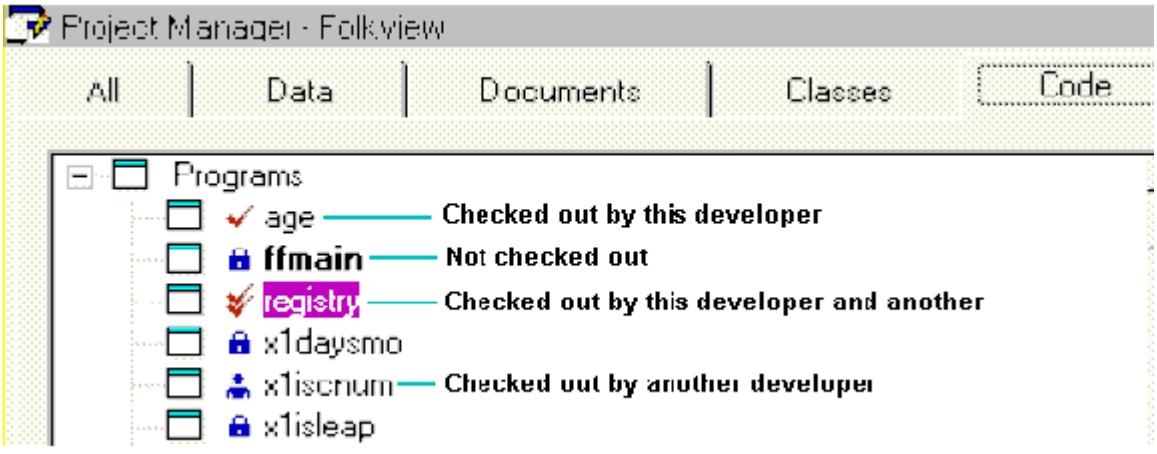

Figure 4: Integrated source code control allows you to see the status of files at a glance.

However, more than a handful of developers have reported problems with the integrated source code control. These problems include:

1. Tying directly into the Project Manager eliminates other development alternatives, like the Component Gallery or the Class Browser
2. Poor performance with larger (n > 2) teams
3. Hassles with file corruption

4. Requirements that multiple checkouts be allowed

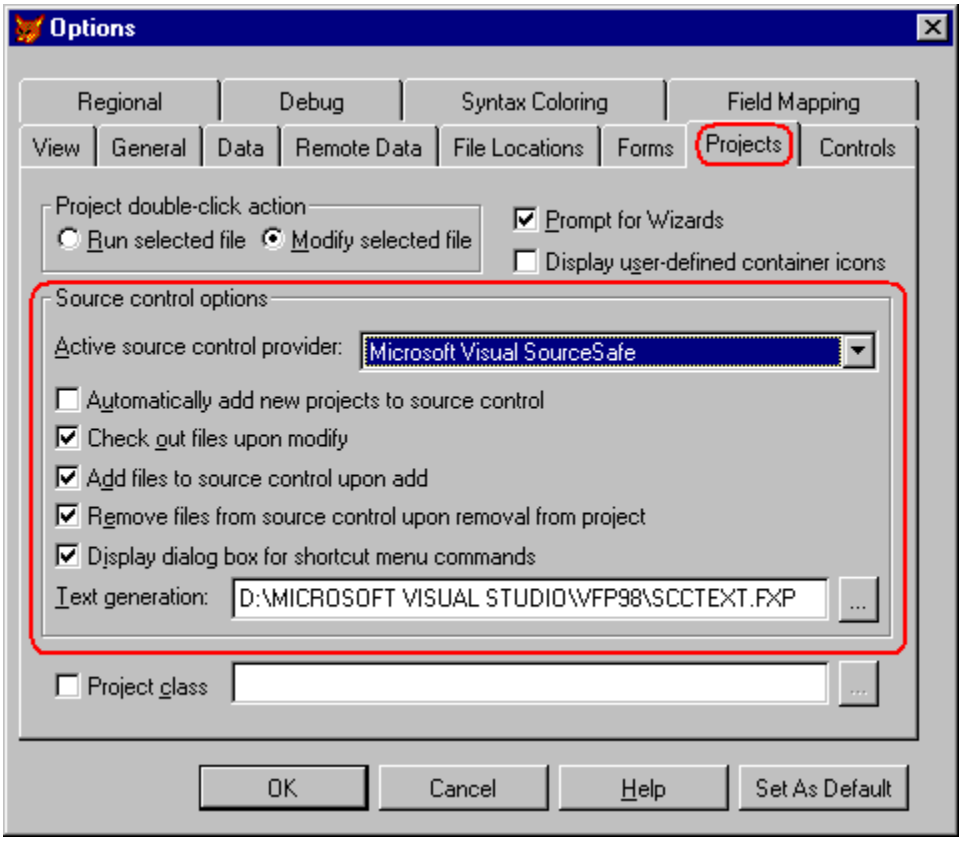

Figure 5: The Tools|Options dialog's Project tab lets you set source code control options.

These are legitimate beefs, and each should be addressed.

2. For the first, it is true that Microsoft only provided the interface to VSS using the Project Manager. However, it is certainly possible for an enterprising developer to develop an add-on for either the Component Gallery or the Class Browser that would also interface with SourceSafe and provide similar functionality.
3. Performance is a major issue with SourceSafe. Over the years, I have seen performance vary wildly depending on the exact version number (and Service Pack), the type of network, the number of protocols used, the depth of the tree within the VSS Project, and other factors. In nearly all cases, I can say that, for the smaller projects (4 or less developers) in which I have been involved, it has been possible to improve performance significantly.
   A second issue with performance is of access over low-speed lines. In earlier versions, VSS slowed to a crawl if you attempted to use a dial-up connection. While the current version has improved performance by orders of magnitude, VSS is still a file-server based system (like VFP when you work directly with DBFs) and will not surprise anyone with its speed. A good solution to consider for low-speed connections is a third-party tool like *Source OffSite™*.
4. There is no question that there is some fragility in the technique of sharing and merging PJM files. Occasionally, a glitch will cause an incorrect update, or the need to rebuild a PJM file manually. However, my experience has been that those infrequent hassles are more than counter-balanced by the occasions when SourceSafe has saved me from myself.

Multiple checkouts seem to be required by the original design of the source code control integration scheme, but it is possible to work around this. I have had reports from several shops that, instead of having everyone check out the PJM (jointly) and then merging, each developer checks out the file, one at a time, before performing an "Update Project List." to resynchronize their project with the master stored in Visual SourceSafe. As best as I can tell, this technique should work as well.

Advanced Topics

Finally, there are a couple of advanced topics that are beyond the scope of what I can cover in this session, but I did want to alert you to the possibilities they provide:

Modifications to SCCTEXT.PRG

The October '97 issue of *FoxPro Advisor* has an excellent article by Mark Wilden with several suggestions on changes to SCCTEXT.PRG. One issue he identified was a problem with SCX and VCX files jumbling the order of methods each time they were saved. When SCCTEXT generated the method code for the corresponding SCA or VCA file, the methods were not sorted, so viewing differences in the files was difficult. Mark proposed a simple change to the SCCTEXT program to sort these methods before writing out the file.
Christof Lange, CompuServe SysOp, FoxPro Advisor Contributing Editor and Microsoft MVP, added his contributions with a cleaner localization (support for foreign languages) in the edition of SCCTEXT included on the conference CD. Thanks to Advisor Publications and Robert Green of Microsoft for permission to reproduce the program, and to the authors for sharing the original ideas with us.

Sharing Files Outside of the Project Tree

Part of the limitations of using integrated source code control requires that all source code elements need to be located within the directory and subdirectories of the location of the project file itself. For those of us who use common code libraries, shared class libraries, and pre-built frameworks and components, this is a real inconvenience. The quick-and-dirty manual answer is to use the SourceSafe interface to share all the components into the project tree in SourceSafe, use "Get latest version" to create a copy on disk, then rebuild the project list, adding all of those items into source code control, and answering the messagebox about "Already under source code control. Delete the local version and get the one in source control?" as "Yes" a few dozen times. I believe that someone has come up with an automated solution to this one, but I have been unable to locate it by press time.

Using Automation

Visual SourceSafe supports Automation to allow you to traverse the project tree, check out, modify, check in, and query the status of each individual tree element. This could be the basis for third-party tools that emulate (or improve!) on the Project Manger integrated source code control. Documentation on the Automation interface is available on the Visual SourceSafe web site at http://msdn.microsoft.com/ssafe.

Third-Party Products

SourceGear Corporation markets a product called SourceOffSite (http://www.sourceoffsite.com) that integrates into your web server and provides client-side access to a SourceSafe database, over the web. Client-side integration provides most of the features available within the SourceSafe product itself, including integration with the IDEs of all of the Visual Studio products. If you have a need for remote access, this is a product I have heard many people rave about.

About the Author

Ted Roche develops LAN, client-server and Web applications using Microsoft Visual FoxPro, Microsoft Visual SourceSafe, SQL Server, and other best-of- breed tools. Recent projects have included some large-scale workflow processing and sales force automation projects with Outlook and Exchange. He runs Ted Roche & Associates, LLC, http://www.tedroche.com/, based in New Hampshire, where the company works with clients nationwide as mentors, teachers and fellow developers. Ted is author of "Essential SourceSafe" (Hentzenwerke Publishing), co-author of the award-winning "Hacker's Guide to Visual FoxPro" series, a contributor to five other FoxPro books and author of numerous magazine articles. Ted is a Microsoft Certified Solution Developer, Microsoft Certified Systems Engineer, and Microsoft Support Most Valuable Professional. Contact Ted at mailto:tedroche@tedroche.com.

References:

SourceOffSite at www.sourcegear.com

Roche, Ted, *Essential SourceSafe,* Hentzenwerke Publishing, 2001. Visit the Hentzenwerke Publishing site (http://www.hentzenwerke.com) for order forms, sample chapters, updates and downloads.