The Fox Hunt

Automate your file opening procedure through a “data-driven” approach

By Whil Hentzen

Many developers open all the files their application needs during the initialization phase of the system, and leave them open until the application is exited. However, the code that is needed is repetitive and boring to write. And it is often needed in several places in the application. It’s easy to automate this process with a generic routine.

First, we’re going to fill a table with the names of all the database in the application. The following routine would automate this process:

```
* f_fill.prg
* creates table named SYSTAB and fills it with the names of
* every DBF in the current directory

create table SYSTAB ;
  (;
  filename  c(12), ;
  aliasname c(12), ;
  fullname  c(25), ;
  tabletype c(5), ;
  status    c(1), ;
  useagain  l(1) ;
  )
=adir(aFiles, "*.DBF")
append fields FileName from array aFiles
return .t.
```

Now that we’ve filled SYSTAB with the file names, we’ll need to populate additional fields in order to tell how to open it. AliasName contains the name of the alias we’ll use when opening the file, TableType contains “SYS” or “DATA”, and Status will contain a code that indicates whether the file should opened with the F_OPENER routine or if it will be opened manually. Finally, the UseAgain field is a logical field that indicates whether or not the file is being opened multiple times.

Next, we’ll use the following routine to automatically open every file.

```
* Program: f_opener.prg
* Purpose: opens every file in the SYSTAB table
*  Syntax: do f_opener with <c Method>
*   Parms: <c Method>
*        : “SYS”   = opens files where TableType = “SYS”
*        : “DATA”  = opens files where TableType = “DATA”
*  Sample: do f_opener with “SYS”
*   Notes: Status = “A” means file is always opened
*        : Status = “M” means manually opened when necessary
*
parameter m.jcMethod

if parameter() < 1
 m.jcMethod = “DATA”
endif
sele FileName, AliasName, UseAgain from SYSTAB ;
  where TableType = m.jcMethod ;
  and Status = “A” ;
  into array aTabToOpen
use in SYSTAB
if _tally > 0
  for i = 1 to alen(aTabToOpen,1)
    wait window nowait “Opening files: “ + aTabToOpen[i,1]
    if file( aTabToOpen[i,1] )
      if aTabToOpen[i,3]
        use &aTabToOpen[i,1] in 0 again alias &aTabToOpen[i,2]
      else
        use &aTabToOpen[i,1] in 0 alias &aTabToOpen[i,2]
      endif
    else
      wait window alltrim(aTabToOpen[i,1] + “not found. Continue/quit (C/Q)?” to j
      if upper( m.j ) = “Q”
        exit
      endif
    endif
```

```
  next
else
  wait window "SYSTAB is empty. Continue/quit (C/Q)?" to j
  if upper( m.j ) = "Q"
    exit
  endif
endif
```

F_OPENER.PRG handles a number of different situations and potential errors so that the file opening process is fully automatic. First, it can be called to open system files (such as data dictonary files) or data files. If it's called without a parameter, the "DATA" parameter is automatically provided. Second, if SYSTAB doesn't contain any files of the appropriate type, or if the file doesn't exist on disk, the user will be prompted whether or not they wish to continue or exit. Third, if the UseAgain flag is positive, the file will be USEd AGAIN.

As a result, we've replaced a number of custom commands that are repeated in various locations throughout a typical application with a single call to a procedure in our common library. Furthermore, this routine can be used throughout the application - for example, in a reindexing routine or in a mechanism to switch data sets.

This routine can be used in all your applications; the only work necessary to adapt it to a new system is to run F_FILL.PRG with the new systems' data files. Note that the F_FILL.PRG program is not as rigorous as F_OPENER.PRG as far as flexibility and error trapping. The public domain utility XCATALOG contains a more robust program that not only creates a SYSTAB file, but will also update it and also populate column and index tables.