February, 2003

Advisor Answers

Sharing menus across versions

VFP 8/7/6

Q: I am currently working in Visual FoxPro 7.0 but have to support Visual FoxPro 6.0 applications. Both my 7.0 and 6.0 applications use the same library (my own) and I have run across a problem I cannot seem to solve.

In my library, I have three shortcut menus that work in textboxes, editboxes and so on. These are popups for spell checking and this kind of thing.

However, when I modify the menu in VFP 7, Visual FoxPro wants to convert these menus to the 7.0 format. I thought about splitting these menus out into two sets, one for 6.0 and one for 7.0, but if I bind these using the VERSION() function in an IF statement or something – won't FoxPro (7.0) still want to convert both types because it will see the reference to both?

Is there any compiler directive or something I could use to maintain the use of my library and still manage to use it in both 6.0 (for the time being) and 7.0, and maybe keep two sets of these menus - and be able to manage this?

–Terry Cunningham (via Advisor.COM)

A: As you note, the format used for menu files (.MNX) changed between VFP 6 and VFP 7. Specifically, VFP 7 includes the ability to specify icons for menu items and the menu format was modified to contain this information.

The good news is that, if you don't open the menu (with MODIFY MENU or through the Project Manager) in VFP 7 or VFP 8, the .MNX file retains the VFP 6 format. That's true even if you build an .APP or .EXE in VFP 7 or VFP 8.

However, I suspect you'd like to do all your new development and modifications in VFP 7 without worrying about the menu format. Fortunately, there's a solution.

The exact steps you need to take depend on whether you want to add icons to the menus for VFP 7 use. I'll show what you need to do if that's not a consideration, and then describe what else is necessary if you do want that ability.

The key to solving your problem is to remember that an .MNX is just a table with a different extension and thus, it can be manipulated programmatically. Combine that capability with a project hook and you should be able to maintain one copy of your menus for both versions.

A project hook is an object that's linked to a project and has methods that fire based on various project events. The event we're interested in here is BeforeBuild, which fires at the beginning of the build process. We can put code there to look at each menu and, if we're in VFP 6, make sure it's in VFP 6 format.

The first thing almost any project hook needs is an object reference to the project to which it's related. Add a custom property to your projecthook subclass called oProject. Then, put this code in the Init method:

```
This.oProject = _VFP.ActiveProject
```

Put this code in BeforeBuild (after the required parameters):

```
LOCAL oFile

IF VERSION(5) <= 600
   * Only do this if we're running in VFP 6 or earlier
   * Replace any menus with copies in the right version.
   FOR EACH oFile IN This.oProject.Files
      IF oFile.Type = "M"
         This.FixMenu( oFile )
      ENDIF
   ENDFOR
ENDIF
```

Add a custom method FixMenu to the class and put this code in there:

```
* Change a menu to VFP 6 format, if necessary
LPARAMETERS oMenuFile

LOCAL aFieldList[1], nFieldCount, lFoundFields

SELECT 0
USE (oMenuFile.Name) ALIAS MenuFile EXCLUSIVE
nFieldCount = AFIELDS( aFieldList )
IF nFieldCount > 23
   * VFP 6 menus have 23 fields
   * Check for the specific fields
   lFoundFields = aFieldList[ 24, 1] = "SYSRES"
   IF nFieldCount = 25
      lFoundFields = lFoundFields AND ;
                     aFieldList[ 25, 1] = "RESNAME"
   ELSE
      * Something's wrong
      lFoundFields = .F.
   ENDIF

   IF lFoundFields
      ALTER TABLE (oMenuFile.Name) DROP SysRes
      ALTER TABLE (oMenuFile.Name) DROP ResName
   ENDIF
ENDIF
USE IN MenuFile

RETURN lFoundFields
```

Save the project hook class, then open the project and use the Project Info item on the context menu or the Project menu to attach the project hook class to the project. A project hook isn't instantiated for a project until the next time you open the project after attaching, so be sure to close the project and re-open it before building.

From this point on, each time you build the project, the menu files are checked and, if necessary, restored to VFP 6 form. This projecthook class is included on this month's Professional Resource CD.

If you'd like to be able to use the new icon capabilities when you're in VFP 7 or VFP 8, you'll need to change the FixMenu method a little bit and add some more code. Rather than just removing the columns, the solution in that case is to make a copy of the MNX and MNT files, then remove the columns. Then, in AfterBuild, you can restore the original versions. I'd be inclined to add an array property to the class to track the list of menu files treated this way. Storing the original name and the name (including path) of the copy for each makes it easy to restore them in AfterBuild.

–Tamar