A Generic About Dialog

Doug Hennig

In the last issue, Doug discussed a couple of dynamic dialogs built with the assistance of Dynamic Form. In this article, he presents another dynamic, generic dialog, this time to display application information.

Most applications have an About function in the Help menu that displays information about the application, such as the name of the company, the version number, the current directory, and so on. As I discussed in the September 2016 issue of FoxRockX, after having created several About dialogs over the years, I decided to create a generic one that dynamically displays whatever settings you wish. While this dialog is data-driven like the ones in the previous issue, it does not use the Dynamic Form VFPX project for control rendering. Instead, it uses a ListView ActiveX control.

Figure 1 shows what the dialog looks like. Everything you see in the dialog except the Copy Settings link and the OK button are customized by passing parameters or adding records to a table. This dialog has the following features:

- The ListView lists as many items as you wish in description/value pairs. The content of the list comes from a table named About which we'll look at in a moment.
- Some of the items in the list appear in blue text indicating they are hyperlinked: clicking them performs some action. For example, clicking the Program Folder item brings up a File Explorer window displaying the contents of the program folder.
- The Copy Settings button copies the items in the ListView to the Windows Clipboard so they can be pasted somewhere.
- The form caption, image, application name, and version number aren't hard-coded but are passed as parameters to the form.
- Resizing the dialog automatically resizes the ListView and its second column. You can manually adjust the size of the columns as you normally would with a ListView.

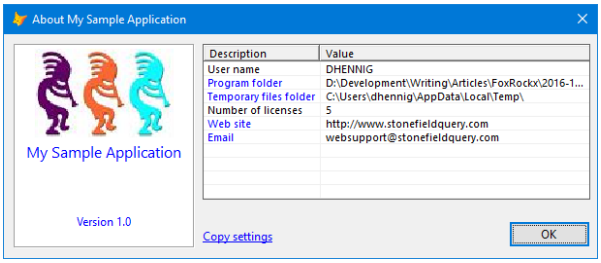

Figure 1. This About dialog is completely generic.

The Init method accepts three parameters: the name of the application, the version number (as a string), and the name of an image to use as the application or company logo. It uses these parameters for the caption of the form and labels for the application name and version number and for the logo image. It then calls the LoadList method to load the settings. Here's the code for Init:

```
lparameters tcAppName, ;
    tcVersion, ;
    tcLogoImage
dodefault()
with This
    .Caption            = 'About ' + tcAppName
    .lblProduct.Caption = tcAppName
    .lblVersion.Caption = 'Version ' + ;
       transform(tcVersion)
    .imgLogo.Picture    = tcLogoImage
    .LoadList()
endwith
```

LoadList starts by opening the About table. This table contains the items to display in the About dialog. **Table 1** shows the structure of this table and **Figure 2** shows the records in the sample table accompanying this article.

Table 1. The structure of About.dbf.

Name	Type	Purpose
Order	I	The order of the item in the dialog
Caption	C(40)	The description of the item
Value	M	The value as an expression to be evaluated
Hyperlink	L	.T. if this item should appear as a hyperlink
Link	M	An expression to evaluate for the hyperlink action; leave it blank to use Value

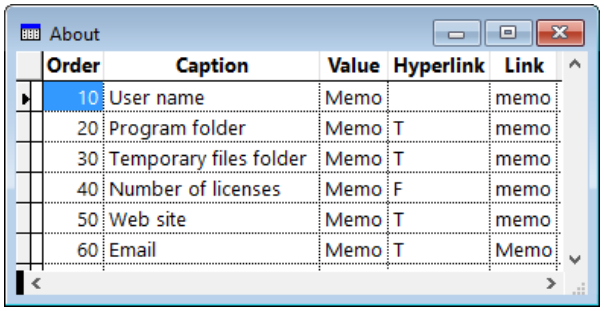

Figure 2. The sample About table.

LoadList goes through the records in the About table to determine the width of the longest description and bumps up that width by 10 pixels to account for spacing in the ListView. It then sets some properties of the ListView control and creates two columns: one for the description of each item and one for the value. Since the value is often wider than the description, the description column only takes up the width calculated earlier and the value takes up the rest of the width of the ListView. Finally, LoadList goes through the records in the About table again, this time adding each item to the ListView by calling the AddSettingToList method, which we'll look at in a moment. Here's the code for LoadList:

```
local lnSelect, ;
    lnWidth, ;
    lnCurrWidth, ;
    luValue
lnSelect = select()
select 0
use ABOUT order ORDER again shared
lnWidth = 0
scan
    lnCurrWidth = txtwidth(trim(CAPTION), ;
        This.FontName, 8)
    lnCurrWidth = lnCurrWidth * ;
        fontmetric(6, This.FontName, 8)
    lnWidth     = max(ceiling(lnCurrWidth), ;
        lnWidth)
endscan
lnWidth = lnWidth + 10

* Set some properties of the ListView.

with This.oList
    .View             = 3
    .LabelEdit        = 1
    .GridLines        = .T.
    .Object.Font.Name = This.FontName
    .Object.Font.Size = 8
    .FullRowSelect    = .T.

* Create some columns for the ListView.

    .ColumnHeaders.Add(, 'Description', ;
        'Description', lnWidth)
    .ColumnHeaders.Add(, 'Value', ;
        'Value',       .Width - lnWidth)
endwith

* Add the settings we want displayed.

scan
    luValue = transform(evaluate(VALUE))
    This.AddSettingToList(trim(CAPTION), ;
        luValue, HYPERLINK, ;
        iif(empty(LINK), '', evaluate(LINK)))
endscan
use
select (lnSelect)
```

AddSettingToList accepts four parameters: the description of the item, the value of the item, .T. if it's hyperlinked, and the value to use for the hyperlink. It checks to see if the item is already in the aItems array of the form and if not, adds a new item to the ListView and the array. It sets the value of the ListView item and if the item is supposed to be hyperlinked, changes the color to blue so it looks like a hyperlink and stores either the hyperlink expression or the value into the Tag property of the item (if the expression is blank, the value is used). Here's the code for AddSettingToList:

```
lparameters tcDescription, ;
    tcValue, ;
    tlHyperlink, ;
    tcLink
local lnRow, ;
    loItem
with This
  if vartype(tcDescription) = 'C' and ;
    not empty(tcDescription) and ;
    len(tcDescription) <= 100 and ;
    not empty(tcValue) and ;
    len(tcValue) <= 255
    lnRow = ascan(.aItems, tcDescription, ;
      -1, -1, 1, 13)
    if lnRow > 0
      loItem = .oList.ListItems(lnRow)
    else
      loItem = .oList.ListItems.Add(, ;
        sys(2015), tcDescription)
      lnRow  = iif(empty(.aItems[1]), ;
        1, alen(.aItems) + 1)
      dimension .aItems[lnRow]
      .aItems[lnRow] = tcDescription
    endif lnRow > 0
    loItem.SubItems(1) = tcValue
    if tlHyperlink
      loItem.Forecolor = rgb(0, 0, 255)
      loItem.Tag       = iif(empty(tcLink), ;
        tcValue, tcLink)
    endif tlHyperlink
  endif vartype(tcDescription) = 'C' ...
endwith
return
```

When the user clicks an item in the list, the ListView's ItemClick method is called. This method checks to see whether the item has anything in the Tag property and if so, uses the Windows API ShellExecute function to open the default application for the item it (ExecuteFile.prg is a wrapper for ShellExecute). So, if the value of the item is a path, File Explorer opens with that path. If it's a URL, the user's default browser opens with that URL. In the case of an email address, use something like "'mailto:' + oApp.cSupportEmail" in the Link field of the About table if oApp.cSupportEmail contains the email address to use.

```
lparameters toItem
if not empty(toItem.Tag)
    ExecuteFile(toItem.Tag)
endif not empty(toItem.Tag)
```

When the form is resized, we want the value column of the ListView to resize as well. The following code in the Resize event of the form takes care of that:

```
local lnWidth
with This.oList
    lnWidth = .Width - ;
        .ColumnHeaders.Item(1).Width
    .ColumnHeaders.Item(2).Width = ;
        max(lnWidth, 100)
endwith
```

Main.prg is a sample program that shows how to use SFAbout. It creates a dummy application object for demo purposes only. SFAbout doesn't use oApp but the records in the sample About table do; for example, the expression for the value of the User Name item is oApp.cUserName. Main then instantiates SFAbout, passing it "My Sample Application" as the application name, "1.0" as the version number, and "KokoWhite.jpg" as the image to use.

```
oApp = createobject('Empty')
addproperty(oApp, 'cUserName', 'DHENNIG')
addproperty(oApp, 'cWebSite', ;
    'http://www.stonefieldquery.com')
addproperty(oApp, 'cSupportEmail', ;
    'websupport@stonefieldquery.com')
addproperty(oApp, 'nLicenseCount', 5)

* Display the About dialog.

loForm = newobject('SFAbout', 'SFAbout.vcx', ;
    '', 'My Sample Application', '1.0', ;
    'KokoWhite.jpg')
loForm.Show()
```

Summary

There are numerous dialogs that most applications have in common, including an Options dialog discussed in the previous issue and the About dialog discussed in this one. Creating generic, dynamic versions of these dialogs means you never have to create them again.

Doug Hennig is a partner with Stonefield Software Inc. He is the author of the award-winning Stonefield Database Toolkit (SDT); the award-winning Stonefield Query; the MemberData Editor, Anchor Editor, and CursorAdapter and DataEnvironment builders that come with Microsoft Visual FoxPro; and the My namespace and updated Upsizing Wizard in Sedna.

Doug is co-author of "VFPX: Open Source Treasure for the VFP Developer," "Making Sense of Sedna and SP2," the "What's New in Visual FoxPro" series (the latest being "What's New in Nine"), "Visual FoxPro Best Practices For The Next Ten Years," and "The Hacker's Guide to Visual FoxPro 7.0." He was the technical editor of "The Hacker's Guide to Visual FoxPro 6.0" and "The Fundamentals." All of these books are from Hentzenwerke Publishing (http://www.hentzenwerke.com). He wrote over 100 articles in 10 years for FoxTalk and has written numerous articles in FoxPro Advisor, Advisor Guide to Visual FoxPro, and CoDe. He currently writes for FoxRockX (http://www.foxrockx.com).

Doug spoke at every Microsoft FoxPro Developers Conference (DevCon) starting in 1997 and at user groups and developer conferences all over the world. He is one of the organizers of the annual Southwest Fox conference (http://www.swfox.net). He is one of the administrators for the VFPX VFP community extensions Web site (http://vfpx.codeplex.com). He was a Microsoft Most Valuable Professional (MVP) from 1996 to 2011. Doug was awarded the 2006 FoxPro Community Lifetime Achievement Award (http://tinyurl.com/ygnk73h).